



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

2006-06

The design and implemtation of a prototype web-portal for the integrated mobile alerting system (IMAS).

Hsu, Michael.

Monterey, California. Naval Postgraduate School

<http://hdl.handle.net/10945/2746>

Downloaded from NPS Archive: Calhoun



<http://www.nps.edu/library>

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**THE DESIGN AND IMPLEMENTATION OF A PROTOTYPE
WEB-PORTAL FOR THE INTEGRATED MOBILE
ALERTING SYSTEM (IMAS)**

by

Phong D. Le
Michael Hsu

June 2006

Thesis Advisor:
Second Reader:

Gurminder Singh
Magdi Kamel

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2006	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE The Design and Implementation of a Prototype Web-Portal for the Integrated Mobile Alerting System (IMAS)			5. FUNDING NUMBERS	
6. AUTHOR(S) Le, Phong D. Hsu, Michael				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) The Integrated Mobile Alert System (IMAS) is a mobile device message alerting system that provides a means for people to stay connected and receive information in a modality that is constantly available to them. The focus of this research is to develop a proof of concept for the common data format and common platform aspect of the proposed architecture. This project concentrates on the design characteristics of the IMAS-portal and server-side database. The thesis determines a way to aggregate, integrate, and sort different message formats in order to be easily displayed on a variety of mobile devices according to user profiles. It demonstrates that a web-portal written in PHP script supported by a relational database is a good configuration for IMAS. Additionally, a proof of concept system that converts messages into disseminated mobile alerts is presented. This thesis marks the founding steps in developing the IMAS.				
14. SUBJECT TERMS Database Technology, Web Design, Mobile Device Technology, Mobile Alerts			15. NUMBER OF PAGES 165	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**THE DESIGN AND IMPLEMENTATION OF A PROTOTYPE WEB-PORTAL FOR
THE INTEGRATED MOBILE ALERTING SYSTEM (IMAS)**

Phong D. Le
Ensign, United States Navy
B.S., United States Naval Academy, 2005

Michael Hsu
Ensign, United States Navy
B.S., United States Naval Academy, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY

from the

**NAVAL POSTGRADUATE SCHOOL
June 2006**

Author:

Phong D. Le

Michael Hsu

Approved by:

Gurminder Singh
Thesis Advisor

Magdi Kamel
Second Reader

Dan Boger
Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Integrated Mobile Alert System (IMAS) is a mobile device message alerting system that provides a means for people to stay connected and receive information in a modality that is constantly available to them. The focus of this research is to develop a proof of concept for the common data format and common platform aspect of the proposed architecture. This project concentrates on the design characteristics of the IMAS-portal and server-side database. The thesis determines a way to aggregate, integrate, and sort different message formats in order to be easily displayed on a variety of mobile devices according to user profiles. It demonstrates that a web-portal written in PHP script supported by a relational database is a good configuration for IMAS. Additionally, a proof of concept system that converts messages into disseminated mobile alerts is presented. This thesis marks the founding steps in developing the IMAS.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	BACKGROUND	1
B.	PURPOSE	3
C.	SCOPE AND METHODOLOGY	3
D.	ORGANIZATION OF THE THESIS	4
II.	BACKGROUND INTRODUCTION OF MOBILE DEVICE TECHNOLOGY	7
A.	MOBILE DEVICES	7
1.	Cellular Phones	7
2.	Pagers	8
3.	Personal Digital Assistants	9
4.	Laptops	10
5.	Summary	11
B.	MOBILE ALERT FORMATS	12
1.	Short Message Service-SMS	12
2.	Multimedia Messaging Service - MMS	13
3.	Instant Messaging - IM	14
a.	POP3	15
b.	IMAP4	15
C.	WIRELESS TECHNOLOGY	16
1.	Global System for Mobile - GSM	16
2.	Personal Communications Services	17
3.	Third Generation - 3G	17
4.	Wireless Fidelity - Wi-Fi	18
III.	IMAS OVERVIEW	21
A.	IMAS	21
1.	System Requirements	21
a.	Mobile Alerts Need to be Seamless	22
b.	Mobile Alerts Need to be Timely and Just in Time	22
c.	Spatial and Temporal Based Alerts	22
d.	Alerts have to be Context-Sensitive	23
e.	Ability to Exchange Information with Desktop Applications	23
f.	Feedback to the Sender	24
g.	User Alert Profiles	24
h.	Advanced Feature to Retrieve More Information	24
i.	Should Automatically Turn On During Emergencies	25
j.	Smart Rendering Technology	25
2.	System Architecture	26
B.	SYSTEM IMPLEMENTATION OPTIONS	26

1.	Operating Systems	27
a.	<i>LINUX</i>	27
b.	<i>UNIX</i>	28
c.	<i>Microsoft Windows NT/2000/2003 Server</i> ...	28
2.	Web Server Software	29
a.	<i>Apache</i>	29
b.	<i>Microsoft IIS</i>	30
3.	Database Management Systems	31
a.	<i>MySQL</i>	31
b.	<i>Microsoft SQL Server 2000/2003</i>	32
c.	<i>Oracle 10g</i>	32
d.	<i>PostgreSQL</i>	33
4.	Scripting Software	34
a.	<i>ASP.NET</i>	34
b.	<i>ColdFusion</i>	35
c.	<i>Perl</i>	36
d.	<i>PHP</i>	37
C.	SYSTEM IMPLEMENTATION SELECTIONS	37
D.	SYSTEM CONFIGURATION	40
IV.	IMAS DATABASE DESIGN	43
A.	INTRODUCTION	43
1.	Planning and Analysis Phase	44
2.	Conceptual Design	45
a.	<i>Data Model</i>	46
b.	<i>Functional Model</i>	46
3.	Logical Design Phase	47
4.	Physical Design Phase	48
5.	Implementation Phase	49
B.	ADDRESSED IMAS REQUIREMENTS	49
1.	IMAS Inputs	52
2.	IMAS Outputs	55
C.	DESIGN SETUP	62
1.	IMAS Server Setup	63
2.	IMAS Web-Portal	63
a.	<i>Design Considerations</i>	63
b.	<i>Web Portal Implementation</i>	64
3.	WebCalendar	66
D.	IMAS DATABASE STRUCTURE	68
1.	IMAS E-R Diagram	69
2.	IMAS Entities	70
3.	Relationships	73
4.	WebCalendar Tables	74
V.	CONCLUSIONS AND RECOMMENDATIONS	77
A.	CONCLUSIONS	77
B.	RECOMMENDATIONS	78

APPENDIX A. IMAS SYSTEM SETTINGS	81
A. MY.INI	81
B. PHP.INI	85
APPENDIX B. IMAS CODE	109
A. CREATE_USER.PHP	109
B. CREATE_USER_SUCCESS.PHP	114
C. EDIT_ENTRY.PHP	117
D. TEST.PHP	133
E. DATABASE TABLES	136
LIST OF REFERENCES	143
INITIAL DISTRIBUTION LIST	147

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	IMAS Framework.....	51
Figure 2.	Create User.....	52
Figure 3.	Calendar View.....	54
Figure 4.	Edit Entry.....	55
Figure 5.	Message Dissemination.....	57
Figure 6.	Message Delivery.....	58
Figure 7.	Meeting Context.....	59
Figure 8.	General Context.....	60
Figure 9.	Do Not Disturb Context.....	61
Figure 10.	SMS Dissemination.....	62
Figure 11.	IMAS Homepage.....	65
Figure 12.	IMAS Database Model.....	70

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Mobile Device Summary.....	12
----------	----------------------------	----

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Dr. Gurminder Singh for the idea, support,
and inspiration for this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

In recent years, mobile devices have become a ubiquitous facet of modern society. The use of devices such as cellular phones, pagers, personal desktop assistants, and laptops is becoming more and more widespread as their supporting communications network infrastructure grows. These devices make it easy for people to connect to friends, family members, and colleagues, etc.

Mobile alerts are information notices constructed in near real-time for forwarding to mobile devices. They may be constructed in the form of simple text messages, multimedia messages, emails, voicemail, or instant messages to various devices.

The use of mobile alerts continues to grow at a pace that matches the use of the mobile devices themselves. With mobile alerts, businessmen and women are able to keep abreast of the latest business developments and decisions made. Professionals such as doctors and emergency responders can be informed of medical emergencies and crisis situations. Military commanders are able to send updates to their forces in tactical situations that do not allow the use of voice transmission. This fact is seen in the recent Iraqi War where mobile devices are used to maintain communication amongst troops deployed in foreign territory.

It is evident that mobile alerts provide users such as military commanders, business professionals, and emergency

responders with a useful application of coordinating Command, Control, and Communications (C3) operations. In cases that demand critical information to be relayed to an end user, a simple phone call can be placed. However, there are times that a user will be located in situations that do not allow for them to receive phone calls. A Marine Corps platoon commander may be in the middle of setting up an enemy ambush. A businessman may be in the middle of a meeting. A student may be in the middle of receiving a lecture. Whatever the restricting situations may be, mobile alerts present a logical solution for sending important information out.

There are technologies and communication infrastructures that exist to support mobile alerts. However, mobile alerting options are so plentiful that they can be overwhelming. A sender is faced with many options. He/she may not know whether he/she should place a cellular-phone call, send an email, or send an SMS message. If the sender chooses the later, that person is faced with the confusion of choosing between which devices to send to. A user can choose from disseminating multiple alert formats such as short text messages (SMS), multimedia messages (MMS), emails, voicemails, and instant messages (IM) to a wide variety of devices.

It is evident that the varying alerting options pose a great problem. Currently, there is no system in place to coordinate the construction and distribution of different mobile alerts. There is no way to ensure that a user receives critical information and updates needed to perform their duties.

B. PURPOSE

The purpose of this thesis is to begin the design of a system that can synchronize various forms of mobile alerts into a common platform. This system, titled the Integrated Mobile Alert System (IMAS), explores the architecture and requirements necessary to enable users to receive any type of mobile alert across all forms of mobile devices. The proposed design will result in a proof-of-concept solution that demonstrates a way for users to specify how they wish to receive alerts throughout differing times within their daily schedules. It will also explore the integration of the techniques used to convert different mobile alert formats with the IMAS platform. In turn, it will be proven that a good solution to meeting IMAS requirements is through the creation of an IMAS web-portal supported by an implemented database.

C. SCOPE AND METHODOLOGY

The scope of this thesis will include: (1) a review of mobile devices, mobile alerts forms, and their supporting technologies, (2) an overview of database, web-portal, web-server, and scripting language solutions needed to implement IMAS, (3) a discussion of IMAS requirements, (5) a thorough exploration of the proposed IMAS database architecture and supporting web-portal, (4) and a complete analysis of the implemented design presented.

The methodology utilized in conducting this thesis research consists of the following steps:

1. Present an introduction to the different mobile devices alerting technologies and formats currently used.
2. Conduct an in depth analysis of supporting IMAS technologies including server operating systems, web server software, scripting languages, and database management systems.
3. Discuss the IMAS requirements.
4. Discuss the database design process.
5. Explain the implemented IMAS configuration
6. Explore the database design architecture in great detail.
7. Conduct a detailed review and analysis.
8. Present the IMAS program scripts.

D. ORGANIZATION OF THE THESIS

The following chapters in this thesis explore various technologies and how they may be implemented to support IMAS. It is organized as follows.

Chapter II provides an introduction to mobile devices, alerting formats, and current wireless technologies.

Chapter III discusses current database technology. This chapter explores the advantages and disadvantages of various operating systems, web-server software, database management systems, and scripting software as well as their employment in IMAS.

Chapter IV presents a detailed explanation of the IMAS database design. The discussion includes a detailed walk-through of the system requirements, design setup, database structure, and integration of the Email Aggregator and SMS Forwarder system.

Chapter V concludes the thesis with conclusions and recommendations.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND INTRODUCTION OF MOBILE DEVICE TECHNOLOGY

A. MOBILE DEVICES

Mobile devices are portable devices that allow people to send and receive information in a mobile manner. They allow users to send and receive information in near real-time. This section discusses some of the most common mobile devices in use today including cellular phones, pagers, personal digital assistants, and laptops.

1. Cellular Phones

Cellular phones have emerged as one of the most widely used communication devices. The cellular phone, commonly referred to as the cell phone, is a mobile device consisting of a transceiver, an antenna, logic/control unit, keypad, and a battery power source. Modern cellular phones include a camera and a LCD screen. Cellular phones communicate via a cellular network of cell sites arbitrated by a master switching center. The master switching center supports trunk lines to the cell sites in the service area. Additional functionalities such as data transfer, multimedia, and text messaging have been added to cellular phones over the years. Technologies typically associated with cellular phones include AMPS, CDMA, GPRS, GSM, TDMA, and UMTS. AMPS, CDMA, GPRS, GSM, TDMA, and UMTS are methods of modulating the radio wave in order to achieve greater bandwidth. Cellular phones are rapidly replacing traditional phones, watches, and dedicated PDA's because

they allow us to communicate whenever and wherever. Newer, multifunctional mobile devices known as the Smartphone combine the advantages of a cellular phone with a small computer. These hybrid devices allow users to check e-mails, surf the web, watch videos, play music, keep track of schedules, take notes, provide GPS navigation, and call others.

The cellular phenomenon creates new social interactions and lifestyles that are rapidly replacing traditional relationships. Cellular phones keep people close. This is good for families living afar. However, it also attaches people close to work.

Cellular phones equipped with cameras or data transfer capabilities pose security threats to corporations and government alike. Some corporation and government buildings have banned the use of cellular devices within their vicinity. Legal applications of cellular phone jammers have been used to restrict cellular phone receptions in certain areas.

Recent studies indicate that using cellular phones while driving increases the risk of vehicle accidents. (Insurance Information Institute 2006) IMAS is created with an intention of addressing information overload problems associated with current cellular and information technologies.

2. Pagers

Pagers are mobile devices that provide one-way or two-way messaging services (Muller 2000, 673). They operate on

FM radio waves and they are tuned to the same frequencies as their paging base stations (Muller 2000, 677). Pagers have been in use since mid-twentieth century to provide a non-obtrusive alert method for doctors (Dayem 1997, 52). The paging infrastructure is small in comparison to cellular infrastructures because FM is used and only text messages are exchanged (Dayem 1997, 52). Pager network cells also cover a greater area than a typical cell site due to worldwide adoption of the FM radio wave standard. A paging service works by sending a page trigger to the central controller which then broadcasts the page to all the local controllers on the network. Pagers work well but require users to reply via a telephone. The recent popularity of cellular phones with text messaging capability has relegated pagers to hospitals and areas where cellular coverage is poor. Technologies typically associated with paging service include two-tone, Flex, POCSAG, and ReFlex (Muller 2000, 679).

3. Personal Digital Assistants

Personal Digital Assistants (PDAs) recently became popular due to advances in electronic miniaturization. A PDA is essentially a very small battery powered computer. It is much smaller than sub-notebooks, or notebook computers that weigh between one to three pounds, and mainstream notebook computers. Apple's Newton MessagePad introduced in 1993 is considered the first true PDA. The first PDA pioneers were rejected by consumers who complained about the excessive weight, poor performance, poor battery life, and unstable software (Muller 2000,

698). The PDA is similar to a traditional and tablet notebook computers with the exception of the keyboard. PDAs usually feature a touch screen with a virtual keypad and a stylus pen for input. All functions are performed with either human fingers or with a stylus. Modern PDAs contain the equivalent processing power as computers did five years ago. They even feature wireless capabilities, removable storage, GPS navigation, and cellular capabilities. These hybrid cellular PDAs are known as Smartphones as discussed under the previous Cellular Phone section. Unlike traditional computers, PDAs feature instant power-off and power-on capabilities by storing the Operating System (OS) in the system ROM. OS updates can be downloaded from the internet and performed by users. PDAs are commonly used by professionals on the move such as delivery personnel, field technicians, medical professionals, and real-estate agents. Typical PDA applications include a calendar, contact list, notepad, and task list. Recent advances in electronic miniaturization combined with popular demand have extended multimedia and wireless capabilities to PDAs. In fact, today's PDAs seamlessly integrate with desktop computers in order to keep the two platforms in sync. PDAs provide users with a very flexible mobile computing solution.

4. Laptops

Laptops are miniaturized computers with a built-in screen and an internal power source. They are commonly referred to as a laptop, a notebook, or a sub-notebook, depending on their size. The first laptops were luggage-

size computers with a small cathode ray tube (CRT) screen and portable weight of twenty two pounds. These luggage-sized laptops used 360k floppy disks for loading programs and for storage. Word processors and spreadsheets were the most popular applications on the first generation laptops. Most modern laptops rival equivalent desktop computers in their features and performance. Most modern laptops feature wireless capability and some even come equipped with 3G cellular capabilities for more mobile connections. PCMCIA/ExpressCard and USB devices allow users to add additional capabilities. Docking stations are also available so users can dock their laptops at home and use them like desktops. In summary, laptops provide users mobile offices without sacrificing connectivity, performance, and productivity.

5. Summary

Table 2.1 below shows the tradeoff between features, mobility, and performance between various mobile devices. Current advances in electronic miniaturization indicate that all mobile devices will eventually integrate with each other. The most notable example is the Smartphone which integrates pagers, cellular phones, and PDAs.

	Mobility	Size	Features	Performance	Battery Life
Cellular Phone	Excellent	Small to Medium	Good	Average	Good to Excellent
Pager	Excellent	Small to Tiny	Poor	Poor	Excellent
PDA	Somewhat to Good	Medium	Good	Good	Average to Good
Laptop	Average	Medium to Large	Excellent	Excellent	Below Average

Table 1. Mobile Device Summary

B. MOBILE ALERT FORMATS

The advent of digital communication allows mobile devices to exchange text and multimedia messages. The most commonly used mobile alert formats are associated with cell phones and computers. These alert formats include Short Message Service (SMS), Multimedia Messaging Service (MMS), Instant Messaging (IM), and electronic mail (Email).

1. Short Message Service-SMS

Short Message Service (SMS) is a phenomenon that grew from the digitalization of telecommunications. Its popularity rivals that of instant messaging. SMS allow users to exchange short text messages consisting of 160 7bit characters or 70 16bit Unicode characters. This does not include the header information which further reduces the number of characters, hence the name short message

service. SMS can be received by users who subscribe to miscellaneous services provided by their cellular provider. It started as an outgrowth of Global System for Mobile Communications (GSM). SMS is similar to e-mail. It stores messages in a server called a Short Message Service Center (SMSC) and forwards the messages to the user's mobile device. The messages are then sent using a 'best effort' algorithm that's similar to a User Datagram Protocol (UDP), an Internet protocol, with the exception that it does not guarantee delivery. Recently many websites also offer SMS text messaging to anyone's cellular phone by using a SMS gateway.

2. Multimedia Messaging Service - MMS

Multimedia messaging service (MMS) is an enhanced version of SMS with the addition that it allows the exchange of multimedia content. The evolution of MMS is due to the greater bandwidth and increased popularity of SMS. SMS was developed for GSM when digital telecommunication first began. However, the introduction of 3G prompted an update to SMS. MMS works on slightly older generation (2.5G) cellular technologies such as GPRS but is much slower in comparison to a 3G capable device. Unlike SMS, MMS has two different delivery methods. MMS can be delivered immediately to the user's mobile device much like SMS' store and forward technique. MMS can also be deferred by storing the messages on a server and letting the user decide what to do with the message.

3. Instant Messaging - IM

Instant messaging (IM) originated as a way for researchers to communicate with each other through a wide area network such as the ARPANET. Its popularity came about due to the need to chat in 'real-time'. Instant messaging differs from e-mail in that messages are not stored but rather forwarded to the users as soon as possible. Modern instant messages also evolved to include capabilities such as video conferencing, voice communication, file transfer, remote access, and encryption. Instant messaging service works by passing information exchange through a server that determines how to deliver the message to the receiver. Some instant messaging services bypass the servers completely and directly connect users to each other. Sophisticated cellular phones and PDAs provide instant messaging capability in addition to SMS. The most popular instant message services used in 2005 include America Online Messaging (AOL) service, Microsoft's MSN instant messaging service, eBay's Skype, and Yahoo Messenger service.

4. E-Mail

E-mail is perhaps the most popular method of exchanging messages between users. In fact, more mail is delivered in electronic form over the Internet than by the US Postal Service (Muller 2000, 244). E-mail emerged in the same way that instant messages started. Unlike instant messaging however, it provides a way for researchers to

both exchange and archive messages. E-mail messages contain both the receiver and the sender information within the header. The messages are typically sent to a mail server using the Simple Mail Transfer Protocol (SMTP). The mail server stores and routes the messages based on the information in the header. There are two popular e-mail access protocols in use today: POP3 and IMAP4.

a. POP3

Post Office Protocol 3 (POP3) was originally designed to support offline message access. A user must download the message before it can be read. Afterwards, the message is deleted from the mail server. The disadvantage with this access mode is that the messages become randomly distributed across all devices.

b. IMAP4

Internet Mail Access Protocol 4 (IMAP or IMAP4) is similar to POP3 except that the messages do not have to be downloaded to be read. Instead, users can manipulate the messages on the server as if the messages were on a local machine. The users also have the ability to download the messages to their own computers. This capability allows multiple devices to access the mail.

C. WIRELESS TECHNOLOGY

Wireless technology refers to the use of radio waves to transmit information. While, early wireless technologies such as AM and FM are still being employed today, most modern wireless technologies are digital and can carry vast amounts of information. Some modern techniques include TDMA, CDMA, QPSK, etc. Wireless technologies have drastically transformed society since its introduction in the late nineteenth century. Today's wireless technologies are used in every aspect of telecommunications from cellular phones, the Internet, broadcasts, GPS, etc. This section covers some of the most popular forms of wireless technology.

1. Global System for Mobile - GSM

The Global System for Mobile (GSM) Telecommunications standard was developed in 1982 to replace different analog systems operating around the world (Muller 2000, 350). GSM was established to provide an international standard of wireless technologies between the different service providers. It initially used to 900 MHz band. GSM is an open, non-proprietary system that uses digital encoding and the time division multiple access (TDMA) technique. (GSM 2006a) It is a family of wireless technology platforms which includes GPRS, EDGE, and 3GSM (GSM 2006b).

2. Personal Communications Services

Personal Communications Services (PCS) provide subscribers with their own tailored communication services such as e-mail, fax, stock quotes, news, etc. PCS is a digital mode of communication and have adopted GSM as its primary mode of transport. PCS networks operate in a similar fashion as the cellular network. However, it is smaller and limited in range. PCS differentiates itself from cellular phone by providing personalized services. Despite their differences, modern cellular networks and PCS share so much in common to the point that there is no longer a clear distinction between them. It should be noted however, that PCS is the first telecommunication standard to enable the concept of universal messaging (Muller 2000, 695).

3. Third Generation - 3G

3G is the current generation of telecommunication technology. It stands for third generation technology. Cellular networks based on 2.5G and 3G both improve upon second generation telecommunication technology by providing the bandwidth needed to deliver multimedia content (Schwartz 2005, 307). The current 3G standards as of present are: CDMA2000, TD-SCDMA, and UMTS. Universal Mobile Telephone System (UMTS) is based on Wideband CDMA. Wideband CDMA is considered as a possible replacement for GSM and is very popular in countries where GSM the infrastructure is strong. CDMA is the preferred method for implementing 3G even though TDMA is still used in some

cases. For example, General Packet Radio Service (GPRS) and Enhanced Data Rates for Global Evolution (EDGE) both use TDMA technology (Schwartz 2005, 334). GPRS and EDGE are in reality part 3G and part 2G. They are sometimes categorized as 2.5G systems. 3G capable devices proved to be extremely popular in countries with an extensive 3G infrastructure. Streaming video, chat, e-mail, file and transfer are just some of the capabilities 3G provides. The 3G infrastructure forms the backbone that provides mobile users access to data whenever and wherever. Laptop computer manufacturers have just recently began to offer 3G compatible wireless cards such as the EV-DO for their products.

4. Wireless Fidelity - Wi-Fi

Wi-Fi is the name for Wireless Local Area network (WLAN) communication devices. It is comprised of several standards such as IEEE 802.11 specifications and eventually IEEE 802.16 specifications. A typical WLAN configuration consists of an access point (AP) and several wireless clients. The clients connect to the AP by an identifier known as an SSID. Encryption and security protocols such as Wired Equivalent Privacy (WEP) and Wi-Fi Protected (WPA) help reduce data hijacking. Wireless LANs provide anywhere between 11MB, 54MB, to several hundred Mega-bytes of bandwidth. The main advantages of Wi-Fi products lie in their mobility, lowered cost, and ease of setup. The main disadvantages are security and limited range. Typical problems involving wireless LAN include shared bandwidth,

high latency, data hijacking, and the restriction of Wi-Fi to local data networks.

THIS PAGE INTENTIONALLY LEFT BLANK

III. IMAS OVERVIEW

A. IMAS

The Integrated Mobile Alert System provides a common method for people to stay connected in order to receive alerts across a wide variety of platforms. Since there are numerous communications technologies in place, it is essential that IMAS includes every alerting technology available in order to ensure mobile alerts reach their intended recipients. The system should use all available channels of communications to deliver incoming alerts to mobile users. IMAS should also provide a means for users to configure their daily schedules in order to set the context of how they wish to receive alerts. For example, an IMAS user in a meeting may only want to receive an SMS text to their cell-phones instead of regular phone calls. That same user may also prefer to be alerted through phone calls when he/she is driving. The following section outlines the requirements IMAS must address in order to transform it into a feasible working system.

1. System Requirements

The following are the key requirements of IMAS listed in the "CS4235 Mobile Devices Project Report," written by Ng Chee Mun:

a. *Mobile Alerts Need to be Seamless*

That is mobile alerts must be available anytime, anywhere and for any device. The mobile alerts solution is not based on a single technology, but a solution consisting of a group of alerts technologies (e.g., SMS, Emails, Internet Messaging, Paging) working in a coordinated and effective way to keep the receiver informed of any incoming contacts or messages that are addressed to him. Mobile alerts must have global access and transaction capabilities which allow users to carry out online transactions.

b. *Mobile Alerts Need to be Timely and Just in Time*

Mobile alerts need to be real time alerts which are sent instantly to receiver when some messages or contacts try to contact the receiver. These are synchronous alerts and will require Quality of Service (QoS) to ensure alerts are received within a short period of time after the messages are sent.

c. *Spatial and Temporal Based Alerts*

Depending on the location and time of occurrence or where the receivers are, context-sensitive alerts send different alerts to the users. Knowledge of the user's current location, as well as history of past locations, can be used to provide additional customized services. For example, if the users are in the city and there are some parades going on the city, these users should receive alerts notifying them of such events. However, such alerts should not be broadcasted to other users leaving or are away from the city. Similarly, if the parade is over, new

users to the city should not be informed of past events (temporal based alerts).

d. Alerts have to be Context-Sensitive

Based on the context in which the users are, the forms of alerts should be appropriate for the occasion. For example, non-intrusive notifications should be used when the users are attending some meetings or seminars. If the users are participating in some outdoors events and carnivals, the alerts should be loud and attention capturing. Similarly, context-based notifications are used in buildings to notify those entering to observe certain behavior (for example if you enter the hospital, you will receive a notification to switch off your hand phone and PDA). Electronic tour guide at national parks and personal guides at museums use static beacons or GPS to locate the users and provide guided explanation to the items in parks or museums. Context notification can also be used in route planning and providing directions for driver to each their destinations.

e. Ability to Exchange Information with Desktop Applications

Mobile alerts should not be confined to peer-to-peer communication among mobile users. They should have the ability to exchange message with desktop computers and state phones lines so that there are bigger connectivity and coverage for the users.

f. Feedback to the Sender

Mobile alerts should provide feedback to the sender so that the sender is aware that the receiver has been informed of his message. This provides some confidence on the system and assurance to the sender and allows the sender to leave appropriate message for the receiver. For example, if the user is attending a meeting and cannot be disturbed, the mobile alert system will inform the sender, so that the sender can leave a message with his contact for the receiver to call back later. Alternately, knowing that the receiver is attending a meeting and may be reached through Internet messaging, the sender can log on into Skype and use textual message to correspond with the receiver.

g. User Alert Profiles

With so many ways of notifying users, there should be an alert system in place that tries all available means to contact people. Users can register with the alert system with their home phone number, cell-phone number, Skype accounts, MSN accounts, etc. Based on the user's priorities, the alert system will run through the list of contacts and try each one until it contacts the user. A similar service is provided by Vonage which provides integrated voicemails, call forwarding, and in-network calls services among.

h. Advanced Feature to Retrieve More Information

Over time the alert solution should move to an interactive format; that is, once alerted, interactive

methods should be utilized to allow the public to seek additional information in the same modality as the original message. For example, an incoming text message on a mobile device could include a prompt for "more" and more information to be called for within the text. An incoming voice message over a mobile phone could also prompt for "more" and more information to be delivered by voice. These feature as well as other interactive technologies can be used to enrich the features of mobile alerts and hence provide more services to the mobile users.

i. Should Automatically Turn On During Emergencies

Emergencies can be a matter of life and death. It is important that emergency alerts reach the person of concern regardless of the mode the receiver configured on his mobile device. Therefore, when emergency alerts are dispatched, all designated receivers should receive them even if their mobile devices are turned off. Mobile devices should have automatic wake up features and continuously polling feature to receive emergency alerts.

j. Smart Rendering Technology

The alert system should intelligently render messages to fit the screen size of all mobile devices. It should format the messages with regards to the device's capabilities and intelligently format messages depending on the requirements of the sending application and capabilities of the receiving devices.

2. System Architecture

The principal requirement of IMAS is to integrate all mobile device alerting technologies into a common platform. In order to accomplish this feat, a service architecture driven by three primary concepts is implemented; first, the Concept of a User Profile needs to be created. This concept allows users to enter personal data, mobile device data, and the specific contexts of their daily schedules into IMAS. When an alert is received, IMAS checks the profile and creates the preferred message format to be sent to that user. Second, the Concept of a Common Platform needs to be implemented. This common platform will be able to process all mobile communications technologies in order to disseminate messages to any mobile device in a timely manner. Third, the Concept of a Common Data Format which stores all alerts for subsequent retrieval and dissemination needs to be explored.

B. SYSTEM IMPLEMENTATION OPTIONS

There are specific system requirements that need to be addressed in order to create IMAS. First, a dedicated computer running a server operating system needs to be used. Second, that same dedicated server needs to have web server software installed. Third, a database management system (DBMS) needs to be implemented in order to manage the IMAS database. Fourth, scripting software needs to be used in order to display IMAS web-content to a web browser. In the following sub-sections, a review of the different

methods and products available to implement in IMAS will be presented.

1. Operating Systems

Operating system (OS) began as simple process control software. Over time, they evolved into modern multi-tasking processing engines. They are intended to manage system resources while a user works with other programs such as word processing applications or robust video games. Modern day operating systems perform a variety of resource management tasks such as background file management, task-switching, thread management, and resource allocation in order to simplify daily computer use for the user. This section introduces some common OSs.

a. *LINUX*

Linux is the most popular open-source operating system. It is popular among computer enthusiasts because its source code can be modified, improved, and redistributed by the developer in nearly any fashion. Many embedded devices feature a lightweight Linux OS which are equipped with the bare essential packages necessary to perform the required functions. Despite Linux's popularity, there are many disadvantages to point out such as the lack of standardization, various versions, and the numerous distributions available. Popular Linux OS distributions available include Novell's SUSE Linux, Mandrake Linux, Fedora Core (formerly Red Hat Linux), and Ubuntu.

b. UNIX

UNIX is an OS known for its reliability, portability, and scalability. The most widely known UNIX system today comes from the Berkeley Software Distribution (BSD) family. These include FreeBSD, NetBSD, OpenBSD, and to an extent, the Mac OS X. Other UNIX distributions include the HP-UX, Sun Solaris, IBM's AIX, and SGI's IRIX. UNIX is typically used by enterprise system administrators, network administrators, and large database administrators. Some UNIX distributions are specifically designed to support certain fields such as design, modeling, and simulation.

c. Microsoft Windows NT/2000/2003 Server

Microsoft's Windows Server OS started in 1991 and launched in 1993 as Windows NT 3.1 Advanced Server. It eventually evolved to the Windows 2000/2003 client and server family. The client side became known as Windows 2000/XP Professional while the server side became known as Windows 2000/2003 Server (Microsoft 2006c). Microsoft Windows 2000/2003 Server features centralized, policy-based management, Microsoft Active Directory service, ASP, COM+, FrontPage Extensions, and XML support. Microsoft's .NET platform is also fully supported on Windows 2003 Server. These features allow MS Windows 2000/2003 Server to manage website, MS Windows 2000/2003 Professional (clients), MS based business applications, shared files, and shared printers all from a common platform. Both the client and server Windows OS feature the same applications, GUI, and

underlying technologies. This tight integration is the key success factor to Windows's growth and popularity. There are currently four versions of Windows 2003 Server (Microsoft 2006c).

2. Web Server Software

Web server is designed to deliver content to requesting clients over the internet and/or intranet. The web server software handles the delivery, request, and storage for all of the content on a website. The deliveries and requests are usually performed on port 80 of the web server machine. Other ports are used to support protocols such as HTTPS, FTP, SSH, etc. Web server software is designed to handle static HTML pages as well as dynamic ASP.NET web applications. Today's modern web server software can support a wide range of scripting languages such as ASP, ASP.NET, ColdFusion, Perl, and PHP. Statistics from Netcraft Web Server Survey Archives illustrates that the four most common web servers in use today are Apache, Microsoft's IIS, Sun's Java System Web Server, and the Zeus Web Server. (Netcraft 2006) At present, the IMAS will only consider employing Apache or Microsoft's IIS to provide web service.

a. Apache

Apache HTTPD Web Server was originally created by Rob McCool and has been managed since 1999 by a non-profit organization called Apache Software Foundation (Apache 2006). Apache is the world's most popular web server

software because it is free. The newest version, Apache 2.0, was rewritten from scratch and does not share any code from its NCSA origins. This new version adds additional flexibility and protocol support. Additional user developed modules greatly expand the functionalities in Apache. Apache can also be used on UNIX OS, Linux OS, Microsoft Windows OS, and Novell OS.

b. Microsoft IIS

Microsoft's Internet Information Service (IIS) is the most popular corporate web server. IIS has always been integrated with Microsoft's Windows NT/2000/2003 Server family OS since its inception. The newest version is IIS 6.0. It is a complete redesign which includes a new fault-tolerant process architecture (Microsoft 2006a). Other improvements include enhanced web infrastructure security, enhanced manageability, more application support, enhanced performance, and improved scalability. For example, Internet Server Application Program Interface (ISAPI) extensions are disabled by default to prevent malicious users from executing unauthorized scripts. IIS shares a common GUI with its Windows family OS and is a service within the Windows NT/2000/2003 Server family OS. Unlike other web server software, IIS is packaged with Windows NT/2000/2003 Server family OS. Some Microsoft Windows desktop OS includes either a personal web server or a limited version of IIS for personal use.

3. Database Management Systems

Database management systems (DBMS) are used to manage data stores which clients can query. IMAS uses a relational database that stores information in a table containing rows and columns. The rows and columns are related to each other through keys known as primary key and foreign key to a relationship. A brief description on database terminology will be provided in chapter IV. Unlike simple flat files stored on data disks, massive amounts of information can be stored and retrieved with relative ease utilizing the Structured Query Language (SQL). This section provides a brief review on current DBMS's available.

a. MySQL

MySQL is the most popular open-source relational database management system that is free, fast, and has a low system requirement (MySQL 2006). It is flexible enough to be used in embedded devices or as a standalone database server. MySQL is typically employed in small to medium sized applications. Its cost and ease of management allow beginners to easily develop their own databases within minutes. It operates on most major OS' such as Linux, UNIX, and Windows. The latest version, MySQL 5.1, features event scheduling, fail-safe replication, partitioning, and XML functions. Despite its many advantage, MySQL lack many properties found in other SQL Relational Database Management Systems (RDBMS) that are only partially remedied with version 5.

b. Microsoft SQL Server 2000/2003

Microsoft's SQL Server is a DBMS typically found in corporations. It was initially based on Sybase SQL until version 7. Since then, Microsoft's SQL Server has evolved from a dedicated database package into a complete enterprise package composed of an enterprise integration tool, reporting server, and OLAP implementation. SQL Server 2000/2005 share a common GUI management console with Microsoft's own Windows family OS. It features replication services, notification services, integration services for data extraction, transformation, and loading (ETL), analysis services for online analytical processing (OLAP), reporting services, integrated management tools, and integrated development tools with Microsoft Visual Studio (Microsoft 2006b). Unlike other DBMS', Microsoft's SQL Server 2000/2005 lacks cross-platform portability and is dependent on Microsoft's Windows OS.

c. Oracle 10g

Oracle RDBMS is another popular enterprise RDBMS. The latest Oracle database version is Oracle Database 10g Release 2. It is the first database designed for grid computing (Oracle 2006a). Grid computing allows a group of low-cost servers to be connected and managed through the Oracle software suite. The grid computing configuration standardizes the equipments, reduces down time (other servers will pick up the requests if one server fails), consolidate member servers and storage devices, and automate day-to-day management tasks with Oracle Enterprise

Manager (Oracle 2006b). Oracle's stored procedures can be written in either Oracle's own database programming language or in Java, one of the fastest growing programming languages on the World Wide Web. The most notable edition is the Oracle Database 10g Express Edition introduced in 2005. This edition is essentially a scaled down free version of the Oracle Database 10g available to both Windows and Linux platforms. It is restricted to 1 processor, 4 GB of data, and 1 GB of memory. Oracle 10g is available on a variety of OS platforms to include Linux, Windows (both 32bit and 64bit), Solaris, HP-UX, and Mac OS X Server.

d. PostgreSQL

PostgreSQL is an open source object-relational database system (PostgreSQL 2006). The major advantage that PostgreSQL offers lies in the fact that it can run stored procedures written with various programming languages such as C/C++, Perl, Java, and Python. Unlike MySQL, PostgreSQL fully supports referential integrity, database transactions, the performance acronym ACID (atomicity, consistency, isolation, and durability). Developers that are faced with the limitation of MySQL often switch to PostgreSQL due to its fully standardized SQL compliance. PostgreSQL can operate on all major OS such as UNIX, Linux, and Windows.

4. Scripting Software

There are two ways to implement scripting languages—client-side and server-side scripting. These scripts perform simple tasks such as keeping a simple count to executing complex programs that allow clients to manipulate photos uploaded to the server. Scripting languages form the logic behind website actions from queries to validating information to the server. Client-side scripting languages rely on the client's browser/computer to run the scripts whose source code can be viewed from the client (Deitel, Deitel, and Goldberg 2004, 690-691). Common client-side scripting languages include JavaScript, Dynamic HTML, ActiveX controls, and Java applets. Server-side scripts, on the other hand, offer more flexibility such as controlling database access, generate dynamic content, server file access, and collect website statistics. Popular server-side scripts include ASP, ASP.NET, ColdFusion, Perl, and PHP.

a. ASP.NET

ASP.NET is a part of Microsoft's .NET platform software family. The main advantage of using ASP.NET lies in the fact that it allows developers to create .NET applications with any .NET compatible language such as Microsoft's Visual Basic .NET, Visual C++ .NET, and C# (Deitel, Deitel, and Goldberg 2004, 744-745). ASP.NET utilizes the .NET platform in order to allow developers to

create multi-tier, database-driven applications without having to learn a new programming language. Another advantage of using ASP.NET lies in its similarity to Microsoft's Visual Basic's usage of controls to simplify website creation. ASP.NET also employs event-driven environments rather than traditional scripting environments such as PHP. It is similar to Java in regards that it was designed to reduce development time and optimize code reuse. There are numerous .NET classes and tools available to ASP.NET developers. Some main disadvantages to ASP.NET include its reliance on Microsoft's Internet Explorer and its specific tailoring to Microsoft's IIS.

b. ColdFusion

Macromedia (now Adobe) ColdFusion is a commercial scripting language used in web development. Similar to ASP.NET, it is more than just a scripting language. Macromedia has integrated many useful features such as PDF conversions, Macromedia Flash integration, ODBC/JDBC integration, cache management, XML parsing, and a GUI management portal throughout the years. One of the strengths of ColdFusion lies in its strong Java integration. Macromedia's Studio suite combines Macromedia Contribute, Dreamweaver, Fireworks, Flash, and FlashPaper to give developers a tightly integrated web development environment. ColdFusion can be readily deployed in nearly any platform or as a Java Application Server. Additional functionalities are added using libraries and extensions. Disadvantages of ColdFusion include its cost, performance, and the limited availability of extensions.

c. Perl

Practical Extraction and Report Language (Perl) is a popular cross platform open-source programming language managed by the Comprehensive Perl Archive Network (CPAN). It integrates the best features from other programming languages such as C, awk, sed, sh, and BASIC and is commonly used for mission critical projects (Perl 2006). Perl also contains an extensive collection of third party modules available on the internet that allows developers to expand its capabilities. Perl is used as a system interface between different systems, a general purpose application development tool, and as an embedded Perl interpreter within other systems. Unlike PHP, Perl is a programming language and difficult to master. It is used extensively by intermediate to advanced web developers. Perl can be employed on many platforms including Linux, Mac OS X, UNIX, and Windows (with ActivePerl). The latest stable release is Perl 5.8.8 (CPAN 2006).

d. *PHP*

PHP Hypertext Preprocessor (PHP) is a very popular scripting language used by websites to serve dynamic web-content. PHP is a scripting language that uses in-line tags within HTML pages to generate content. It was initially developed by Rasmus Lerdorf and called Personal Home Page Tools. Since then, PHP has been rewritten with the Zend Engine. There are two versions of PHP. The current version is PHP4 and the next generation version is PHP5. PHP5 uses the Zend Engine 2 and incorporates numerous object oriented programming features (PHP 2006). One major benefit of using PHP lies in its ease of use and its extensive support across almost every major operation system. PHP also incorporates extensive interfaces to popular RDBMS such as IBM's DB2, Microsoft SQL Server, MySQL, Oracle Database, and PostgreSQL. It is cross platform compatible and can be operated on Linux, Mac OS X, UNIX, and Windows machines. PHP is easy to learn, use, and deploy on nearly any machine.

C. SYSTEM IMPLEMENTATION SELECTIONS

This section covers the IMAS implementation process. It discusses the systems and software used to set up and configure IMAS.

First, the system implementation selection is determined by the cost, ease of management, learning curve, and performance of the available products. Based on the software research conducted, the most common web server combination utilizes the LAMP configuration. LAMP stands

for Linux, Apache, MySQL, and Perl/PHP/Python. IMAS utilizes the LAMP configuration as a guideline for implementation.

IMAS developers tested each of the three OS platforms and concluded that Microsoft Windows 2003 Server would be the OS of choice. SUSE Linux 9 Professional (FTP), Mandrake Linux 10 Community Edition, Fedora Core 3, FreeBSD 5.1, and Windows 2003 Server were installed over a one week period to test each platform. All the platforms are open-source and free with the exception of Microsoft Windows 2003 Server. FreeBSD 5.1 is the hardest to configure and setup. It required numerous tweaking just to boot to the GUI. Linux distributions fared much better. However, most Linux web development packages lack the ease and intuitiveness that Windows based web development package offer. The lack of new drivers and standardization plague Linux and Unix distributions deeply. Microsoft Windows 2003 Server was licensed under MSDNAA and available to the developers free of charge. Its familiar GUI, management console, and environment offered the best compromise in cost, ease of management, learning curve, and performance.

Initial software tests indicated that both Apache 2.0 and IIS 6.0 would work fine with IMAS. All the Linux and Unix distributions came with the Apache package and can be enabled with ease. Apache for Windows was downloaded from the Apache Software Foundation website. IIS 6.0 was installed with Windows 2003 Server. Additional changes and modifications to the default installation of Apache proved to be troublesome. It requires much more technical expertise than IIS 6.0's intuitive management console. Apache 2.0 and IIS 6.0 proved to be equally apt in other

sectors of concern. Microsoft IIS 6.0 is used primarily due to its ease of management and its integration with Windows 2003 Server.

IMAS developers only tested two RDBMSs; MySQL and Oracle Database 10g Express Edition. PostgreSQL requires a higher learning curve and technical knowledge to setup than MySQL so it was not considered. Microsoft SQL Server 2005 was not part of the MSDNAA license agreement with the Naval Postgraduate School at the time IMAS implementation took place. Thus, it was not tested. IMAS is particularly interested in the possible use of Oracle Database 10g Express Edition due to its power and robustness. MySQL 5.0.18 and Oracle Database 10g Express Edition are both easy to configure and setup. Oracle Database 10g Express Edition has a slight higher learning curve than MySQL but it does come with a powerful management tool. In the end, the authors decided to employ MySQL 5.0.18 because it can be rapidly deployed and managed. Tools such as MySQL Administrator 1.1, MySQL Query Browser, and phpMyAdmin 2.7 significantly reduced the IMAS development time.

IMAS tested three scripting languages: ASP.NET, ColdFusion MX 7(Developer Edition), and PHP. Perl was not tested because it requires developers to learn a new programming language. Microsoft Visual Studio 2005 and Dreamweaver 8 provide a very extensive integrated web development environment for IMAS. ASP.NET requires an extensive learning curve to use. It is not used due to its lack of cross platform portability. ColdFusion proved to be just as easy to install and use as PHP. ColdFusion's major deficiencies are its large memory footprint (especially with JRun) and sub-par performance. IMAS uses

PHP because it supports in-line tags, its cost, its average learning curve, and decent performance.

IMAS was implemented under the purview of the developer's design experience and available software for use. In the end state, IMAS employs the WIMP (Windows, IIS, MySQL, and PHP) configuration even though there are numerous OS', databases, and scripting languages available to the developers. The current IMAS does not require any expensive enterprise level features. In the future however, cost reduction can be further achieved by migrating IMAS onto a Linux or UNIX machine running Apache web server. As of present, the current IMAS version mostly employs open-source software that is readily easy to deploy, learn, and use.

D. SYSTEM CONFIGURATION

The specific system configuration for IMAS is described in detail in Appendix A and is comprised of:

- Dell Optiplex GX270 running Pentium 4 (3.2 GHz with hyperthreading), 2 GB of RAM and 30 GB of hard disk space.
- The hard disk is formatted in NTFS as a single partition.
- Microsoft Windows 2003 Server Standard Edition with IIS 6.0 is installed and updated to the latest patches as of April 17, 2006.

- Microsoft IIS 6.0 is configured to use `C:\inetpub\wwwroot` as default website root directory.
- IMAS is placed under `C:\inetpub\wwwroot\IMAS`. The `index.htm` file in the default website root directory is created to redirect all traffic to IMAS.
- MySQL 5.0.18 is installed and configured as a multi-use database. It is not configured as a dedicated transaction database server.
- MySQL is set to handle 160 concurrent connections, configured to use InnoDB, and uses the default TCP port of 3306.
- Remote root access to MySQL is disabled.
- The default character set in MySQL is set to UTF8 and old password encryption is enabled.
- MySQL 5's new password encryption created severe problems during login purposes.
- Login troubleshooting centered the problem to MySQL 5's new password encryption feature. The old password feature allows MySQL 4.0 or older clients to connect to the database.
- PHP is installed using the PHP EasyWindowsInstaller 4.3.10.2. (PHP EasyWindows Installer 2005)
- This installer was used due to Microsoft IIS 6.0's enhanced security which made normal PHP configuration difficult.

- PHP 5.1.2 was originally installed but proved to be too challenging to configure under IIS 6.0.
- Microsoft IIS 6.0's script settings were modified to allow all ISAPI extensions to allow running PHP scripts.
- FastCGI and Turck MMCache were enabled to enhance performance.

IMAS also employs the following support tools to develop IMAS:

- MySQL Administrator 1.1
- MySQL Query Browser 1.1
- phpMyAdmin 2.7.0-pl2
- phpMyAdmin is installed in *C:\inetpub\wwwroot\IMAS_phpMyAdmin.*
- DBDesigner4 by fabFORCE.net is used to model the database relationships.
- The primary web development tools used are IDM Computer Solutions' UltraEdit-32 Text/Hex Editor version 11.20, Macromedia Dreamweaver 8, Macromedia Fireworks 8, and Macromedia Flash 8.
- All IMAS diagrams are designed using Microsoft Office Visio 2003.

IV. IMAS DATABASE DESIGN

A. INTRODUCTION

The core purpose of the Integrated Mobile Alerting System is to collect information about various clients' mobile devices and their daily schedules in order to disseminate alerts in the most timely and appropriate manner. In order to accomplish this, a database processing system was used. The primary function of a database system is to collect, sort, organize, and retrieve records about physical and logical entities. The database processing system itself consists of an end-user, a database application, a database management system (DBMS), and a database. Before the IMAS database design is explored, it is paramount to become familiar with basic database terminology. The following definitions listed below were taken from the 9th edition of "Database Processing," written by David M. Kroenke:

- *Entity/Table*- Something of importance to a user that needs to be represented in a database.
- *Attribute*- A field or column that helps describe an entity's characteristics.
- *Primary Key*- A uniquely identifying attribute of an entity.
- *Composite Key*- A key with more than one attribute.

- *Foreign Key*- An attribute that is a key of one or more relations other than the one it appears in. Foreign keys are used to represent relationships.
- *Relationship*- An association between two entities.

In order to meet IMAS user requirements, the five steps of database design were considered. These five steps are:

1. Planning and Analysis
2. Conceptual Design
3. Logical Design
4. Physical Design
5. Implementation

The remainder of this section will be devoted to explaining how the IMAS database processing system was developed under the purview of the five design steps. In turn, the reader will be well informed of exactly how IMAS processes records about client mobile devices and their daily schedules in order to generate appropriate corresponding alert types.

1. Planning and Analysis Phase

The initial planning and analysis phase addresses how the database processing system will meet user requirements. It specifically defines what the database is expected to do. The goals of requirements analysis are:

- 1) To determine the data requirements of the database in terms of primitive objects.
- 2) To classify and describe the information about these objects.
- 3) To identify and classify the relationships among the objects.
- 4) To determine the types of transactions that will be executed on the database and the interactions between the data and the transactions.
- 5) To provide the rules governing the integrity of the data (Ward 2001, 8).

There are many ways in which requirements can be produced. Some common methods include conducting interviews, examining existing forms, generating reports from studies, and modifying preexisting databases. In this case, the IMAS requirements were already captured in Ng Chee Mun's "CS4235 Mobile Devices Project Report."

2. Conceptual Design

A conceptual model is a database designer's "mental map" of the system he/she wishes to model. It represents how the information they are encountering should fit together. The conceptual model considers the end-users view and how they will use the system. It provides enough information for the designers to construct a working logical model while avoiding bogging down the end-user with details. In developing a conceptual model, both a data model and a functional model are developed.

a. Data Model

A data model serves as the conceptual blueprint of how the data structures of the database are represented. Three issues are addressed in the data model: 1) the data objects, 2) the associations between data objects, and 3) the rules that govern operations on the objects. The data model serves as the bridge for translating user requirements into a working logical model. Generally speaking, there are two methodologies a designer may consider when developing a data model; the Entity-Relationship (E-R) model and the Semantic-object model (Kroenke 669). The main differences between both methodologies is that the E-R model consists of a set of graphics and symbols to represent entities and their associations while the Semantic-object model incorporates objects that contain relationships in the users world. The IMAS system uses the E-R model since it is more common across more database products and processing platforms.

b. Functional Model

While the data model defines the entities of the user's world, the functional model performs operations on the tables itself. Results from the planning and analysis phase are addressed in this phase. More specifically, the question, "What do you want this database to provide you?" is primarily answered with queries at this stage (Ward 2001, 11). Query operations include adding, deleting, and updating tables in the database as well as retrieving information to generate reports.

The most common language to query tables in a relational database is called Structured Query Language (SQL). SQL was developed by the IBM Corporation in the late 1970's. It was also endorsed by the American National Standards Institute (ANSI) in 1992 and considered to be the industry standard on query language (Kroenke 2004, 192). The IMAS performs its web-portal operations through the use of SQL statements embedded inside PHP code.

3. Logical Design Phase

Following the successful creation of a conceptual model, a logical model is derived. The E-R diagram is refined through a process called normalization. Normalization is the "process of evaluating a relation to determine whether it is in a specified normal form and converting it into those specified normal forms if necessary (Kroenke 2004, 675)." A well planned normalized database meets the following criteria:

- 1) *A good data model is simple.* The data attributes that describe an entity should only describe that entity.
- 2) *A good data model is essentially non-redundant.* Each data attribute, other than foreign keys, describes at most one entity.
- 3) *A good data model should be flexible and adaptable to future needs* (Whitten 2004, 559).

In order to fulfill normalization requirements, a series of normalized forms need to be briefly discussed. In the simplest form, first normal form (1NF) is defined as

an entity whose attributes have no more than one value for that entity instance. In other words, each entity attribute contains only one value. If more than one value exists, then another entity table must be created. An entity is in second normal form (2NF) if it is already in 1NF and its nonprimary-key attributes are dependent on the full primary key. This is especially important to consider in entities that contain composite keys. Finally, third normal form (3NF) exists when an entity is already in 2NF and transitive dependencies, or the situation that exists when all nonprimary-key attributes are not dependent on any other nonprimary-key attributes does not exist. The IMAS database follows all the rules and conventions required to meet 3NF to be a fully normalized database.

4. Physical Design Phase

The physical design aspect of a database is the actual translation of the planning and analysis into a system model that depicts the technical implementation of the user's requirements (Whitten 2004, 102). Key considerations with physical design include the optimization of performance. A good physical design minimizes the number of input/output transfers and efficiently uses external and virtual storage. Most modern database management products come pre-tuned to optimize performance through their processing and storage efficiencies. Since this is the case, the IMAS project does not deeply investigate performance optimization issues.

5. Implementation Phase

The final stage of database design involves the actual creation of the logical and physical model into a working database. The implementation phase considers what database design products to use and how they need to be configured in order to meet user requirements. User interfaces are created as forms while database queries are used to input/delete/update the database. The user should be able to generate reports that extract the specific information he/she needs. In the following sections, a thorough discussion on how the requirements were translated into the implemented IMAS database will be presented. More specifically, the configuration of the ER diagrams, user forms, IMAS tables, queries, reports, and database management software will be explained in great detail.

B. ADDRESSED IMAS REQUIREMENTS

IMAS addresses the planning and analysis phase of database design in the fact that specific system requirements are met. It addresses the following requirements listed in Chapter II with the following solutions:

- 1) *Mobile alerts need to be seamless-* The IMAS database contains records of all user mobile devices, their account information, and providers so that alerts are not restricted to limited technologies.

- 2) *Mobile alerts need to be timely and just in time-*
A proof of concept program that converts incoming email messages to cellular phone SMS text titles *test.php* is integrated with the IMAS. The implementation of this program with IMAS proves that it is possible to retrieve messages from a data store and disseminate them to mobile devices in near real time.
- 3) *Alerts have to be context-sensitive-* The IMAS web portal is designed with customizable user calendars where a user can input the events of the day as well as the context of that particular event. This feature leads to the dissemination of a message to the most appropriate device based on the context setting and alert type desired.
- 4) *User alert profiles-* The IMAS web portal allows users to create accounts which inputs the user's name, login name, password, their mobile devices, mobile provider, email address, instant message screen-names, and instant-message providers. Newly created users will be able to login to the IMAS and customize their profiles with the online calendar.

The following figure illustrates a high-level depiction of the IMAS resulting from the planning and analysis phase.

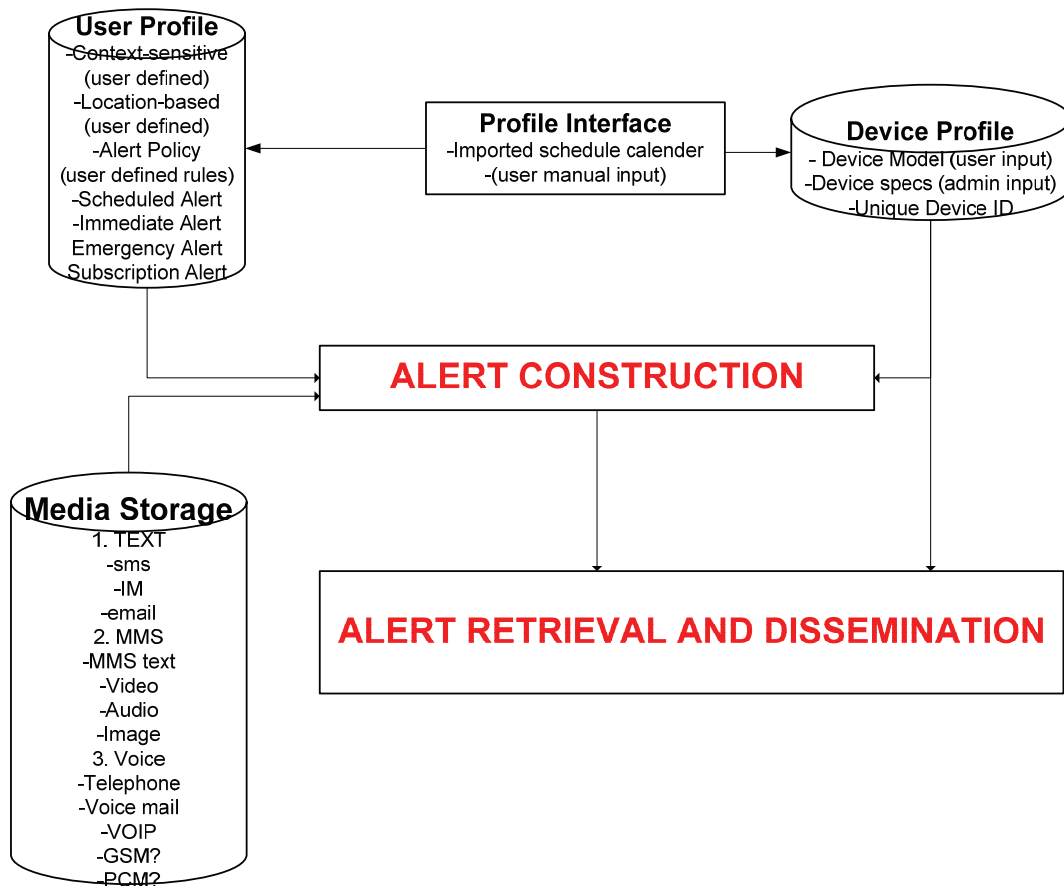


Figure 1. IMAS Framework

In this figure, an IMAS user enters the web portal system through the *profile interface*. The *user profile*, which contains the context and alert type of the user event is entered or updated in the system. According to the parameters the user set in the *user profile*, when an alert comes in through the *media storage*, the IMAS checks the settings that the user defined in the profile, pulls mobile device information from the *device profile*, and constructs an alert in *alert construction*. After the alert is made, the system finally sends the message out thorough the *alert retrieval and dissemination* stage.

1. IMAS Inputs

Before an IMAS user can begin using the system, he/she must create a user account. In the scripting page labeled *create_user.php* listed in Appendix B, the user enters their first name, last name, desired user name, an encrypted password, an email address, email password, IM screen-name, IM provider, cell-phone number, cell-phone provider, and cell-phone model into the system. Once the user presses "submit," the information is entered into the appropriate tables in the IMAS database. The following figure is a screen capture of *create_user.php*.

The screenshot shows the 'Integrated Mobile Alert System' (IMAS) web interface. At the top, there is a banner with the text 'Integrated Mobile Alert System' and a small logo. Below the banner, there is a navigation bar with links: 'Home', 'Login', and 'Create New Account'. The main heading is 'Create new IMAS user account'. Below this, a instruction says 'Please fill out the fields below.' The form contains several input fields: 'First Name:', 'Last Name:', 'Login Name:', 'Password:', 'Email:', 'Email Password:', 'IM Screen-name:', 'IM Provider:' (with a dropdown menu showing 'AOL'), 'Cell # (10 digits only):', 'Cellphone Provider:' (with a dropdown menu showing 'Cingular'), and 'Cell Model:' (with a dropdown menu showing 'Ericsson m45'). There are 'Submit' and 'reset' buttons at the bottom of the form. At the very bottom, there is a footer with the text 'IMAS v0.2 (1 Mar 2006)' and a link to 'About Us | Site Map | Privacy Policy | Contact Us | ©2006 Naval Postgraduate School'.

Figure 2. Create User

The following SQL statements address the functional model in the conceptual phase and are used to enter the user's profile as well as device profile information:

- Inserts user first name, last name, login name, password, and email into the webcal_user table.

```
$insert_into_webcal_user="INSERT INTO
webcal_user
```

```
(cal_firstname,cal_lastname,cal_login,cal_passwd,
cal_email) VALUES
(\"$fname\", \"$lname\", \"$login\", \"$pword\", \"$e
mail\");$into_webcal_user=mysql_query($insert_in
to_webcal_user, $IMASInternalDatabase) or
die(mysql_error());
```

- Inserts user email address, login name, email password, IMAP server name, and email login name into email_account_tracking table.

```
$insert_into_email_account_tracking= "INSERT INTO
email_account_tracking(email_address, cal_login,
email_password, email_imap, email_name) VALUES
(\"$email\", \"$login\", \"$email_pwd\", \"$email_im
ap\", \"$email_login\");
```

- Inserts user screen-name, IM provider, and login name into im_account_tracking table.

```
$insert_into_im_account_tracking= "INSERT INTO
im_account_tracking
(Screen_name,IM_provider,cal_login) VALUES
(\"$im_sn\", \"$im_provider\", \"$login\");
$insert_into_im_account_tracking =
mysql_query($insert_into_im_account_tracking,
$IMASInternalDatabase) or die(mysql_error());
```

- Inserts user's cell-phone number, cell-phone device type, cell-phone provider, and login name into cellphone_account_tracking table.

```
$insert_into_cellphone_account_tracking= "INSERT
INTO cellphone_account_tracking
(cellphone_number,device_brand_number,
cell_provider,cal_login) VALUES
(\"$cellphone\", \"$cellbrand\", \"$cellprovider\",
 \"$login\");
$insert_into_cellphone_account_tracking =
mysql_query($insert_into_cellphone_account_tracki
ng, $IMASInternalDatabase) or die(mysql_error());
```

When a user logs into the IMAS, he/she will be presented with a default calendar view of the current week as illustrated in Figure 3. From this point, the IMAS user can enter his/her personal events into the system. The

event entry form page, labeled *edit_entry.php* and listed in Appendix B, inputs basic event information such as a brief description, full description, date, time, duration of event, context and alert type of the event in the system. *Edit_entry.php* is illustrated in Figure 4.

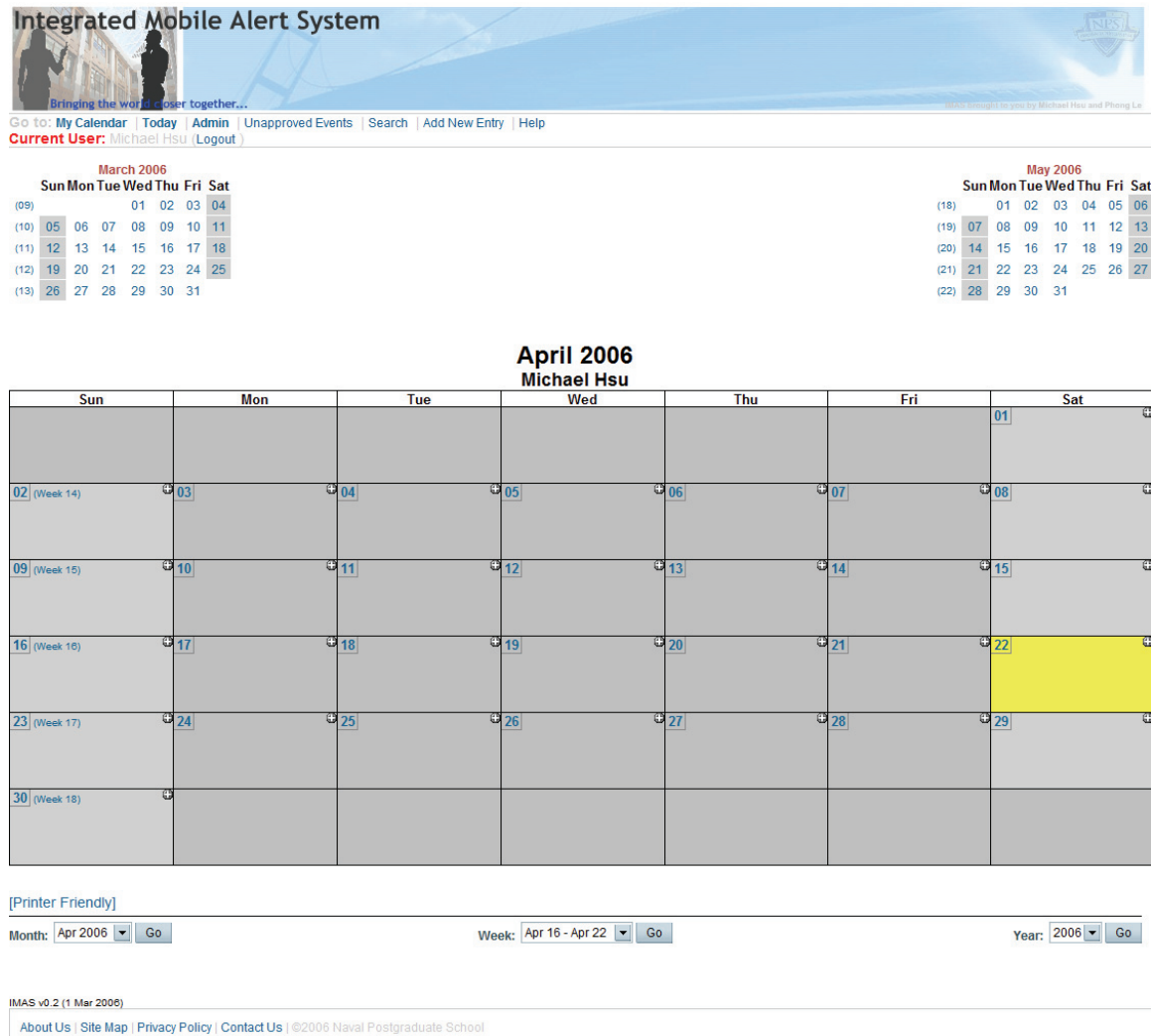


Figure 3. Calendar View

Integrated Mobile Alert System

Bringing the world closer together...

Go to: [My Calendar](#) | [Today](#) | [Admin](#) | [Unapproved Events](#) | [Search](#) | [Add New Entry](#) | [Help](#)

Current User: Michael Hsu | [Logout](#)

[Add Entry](#)

Details | [Participants](#) | [Repeat](#)

Brief Description:

Full Description:

Context:

Alert Type:

Date:

Send Reminder: ☐ Yes ☒ No days hours minutes before event

Month:

Week:

Year:

IMAS v0.2 (1 Mar 2006)

[About Us](#) | [Site Map](#) | [Privacy Policy](#) | [Contact Us](#) | ©2006 Naval Postgraduate School

Figure 4. Edit Entry

2. IMAS Outputs

This section describes the outputs that IMAS delivers and can be viewed as the result of the logical design model. The system functions as follows: When a new alert arrives at IMAS, the system processes the message via the alert construction phase. Afterwards, it sends the message out to the intended recipient based on the context and alert type preference the recipient set for that particular time. Figure 5 and 6 displays the entire process of message arrival to IMAS to message dissemination for a specific user. The "action" entity in the IMAS database uses a composite primary key, `cal_access` and `cal_priority`, which respectively corresponds to context and alert type to extrapolate the appropriate action taken. For our

purposes, only immediate alert type messages, denoted as a "1," is considered as location, subscribed, and scheduled alert types were beyond the current purview of the project. The three context types that IMAS currently processes are general, meeting, and do not disturb." In a general context, the user will receive a SMS message, IM, email, and a cell-phone call. During a meeting, a SMS message, IM, and email will be sent. In a do not disturb context, an email is the only alert disseminated. Figures 7, 8, and 9 displays the alert dissemination to be taken based on the varying contexts. Please note the differing "Alert Notification Methods."

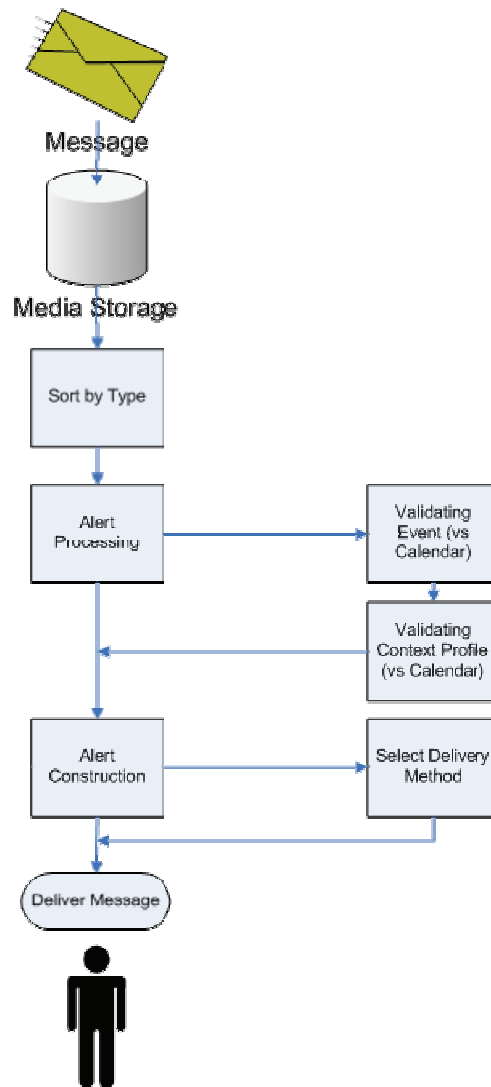


Figure 5. Message Dissemination

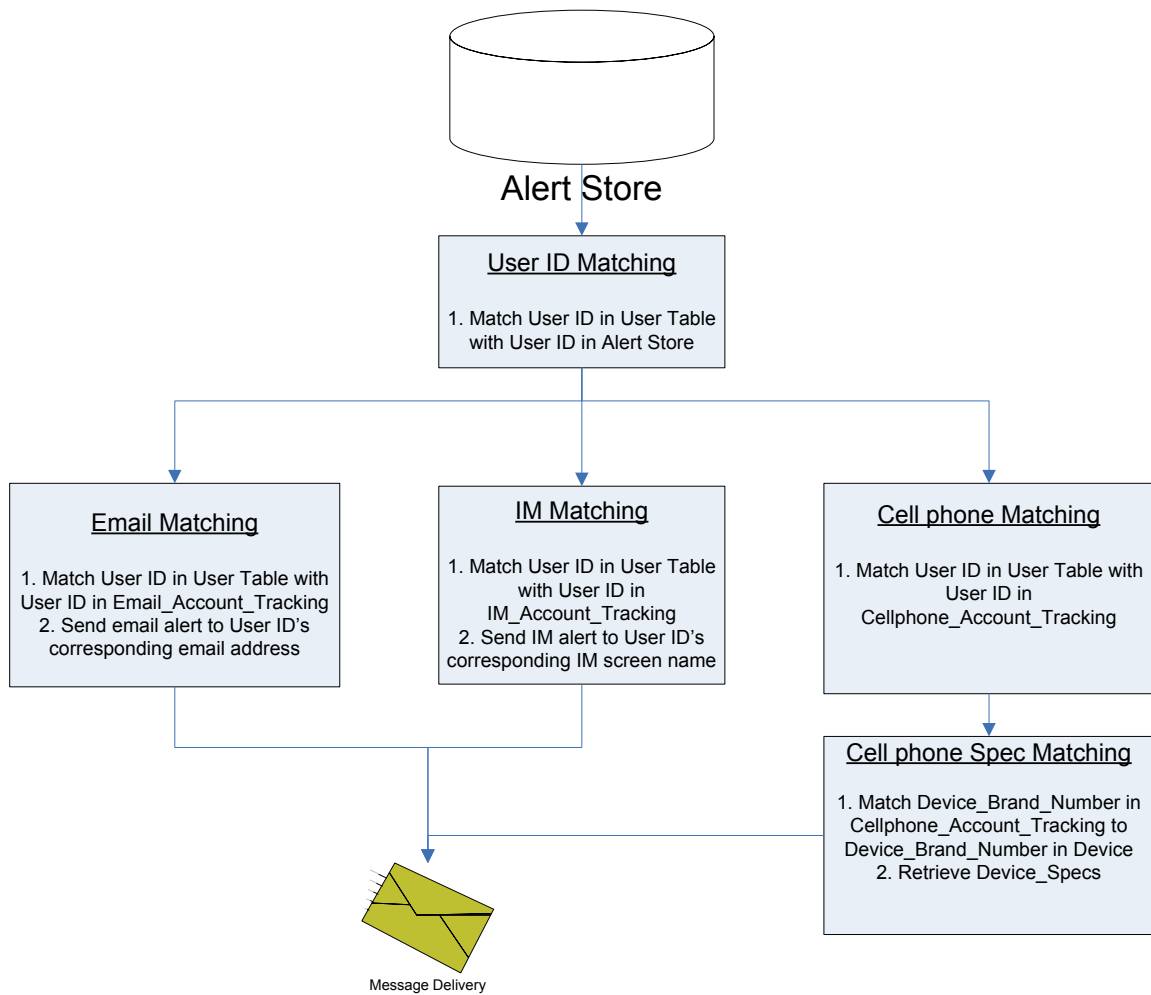




Figure 6. Message Delivery



Integrated Mobile Alert System

Bringing the world closer together...



Alerts brought to you by Michael Hsu and Pheng Li

[Go to: My Calendar](#) | [Today](#) | [Admin](#) | [Unapproved Events](#) | [Search](#) | [Add New Entry](#) | [Help](#)

Current User: [Michael Hsu](#) ([Logout](#))

Meeting

Description: Meet with Professor

Date: Wednesday, April 26, 2006

Time: 14:00-15:00

Duration: 60 minutes

Alert Type: Immediate

Context: Meeting

Created by: [Michael Hsu](#)

Email: mhsu@nps.edu

Cellphone Number: 8018152413

IM provider: AOL

IM screen-name: RunnerXBest

Alert Notification Method: cellphone message,IM,email

Updated: Wednesday, April 26, 2006 15:20

Participants: [Michael Hsu](#)

[Printer Friendly](#)

[Edit entry](#)
[Delete entry](#)
[Copy entry](#)
[Email all participants](#)

Month:


Week:

Year:

IMAS v0.2 (1 Mar 2006)


[About Us](#) | [Site Map](#) | [Privacy Policy](#) | [Contact Us](#) | ©2006 Naval Postgraduate School

Figure 7. Meeting Context



Integrated Mobile Alert System

Bringing the world closer together...



Alert brought to you by Michael Hsu and Pheng Li

[Go to: My Calendar](#) | [Today](#) | [Admin](#) | [Unapproved Events](#) | [Search](#) | [Add New Entry](#) | [Help](#)

Current User: [Michael Hsu](#) ([Logout](#))

Meeting

Description: Meet with Professor

Date: Tuesday, April 18, 2006

Time: 13:00-14:00

Duration: 60 minutes

Alert Type: Immediate

Context: General

Created by: [Michael Hsu](#)

Email: mhsu@nps.edu

Cellphone Number: 8018152413

IM provider: AOL

IM screen-name: RunnerXBest

Alert Notification Method: cellphone message,IM,email,cellphone call

Updated: Wednesday, April 26, 2006 15:10

Send Reminder: Yes - 4 hours before event

Participants: [Michael Hsu](#)

[Printer Friendly](#)

[Edit entry](#)
[Delete entry](#)
[Copy entry](#)
[Email all participants](#)

Month: [Apr 2006](#) [Go](#)

Week: [Apr 16 - Apr 22](#) [Go](#)

Year: [2006](#) [Go](#)

IMAS v0.2 (1 Mar 2006)

[About Us](#) | [Site Map](#) | [Privacy Policy](#) | [Contact Us](#) | ©2006 Naval Postgraduate School

Figure 8. General Context

Integrated Mobile Alert System

Go to: [My Calendar](#) | [Today](#) | [Admin](#) | [Unapproved Events](#) | [Search](#) | [Add New Entry](#) | [Help](#)
Current User: Michael Hsu ([Logout](#))

Exam

Description: Exam in Electrical Engineering.
Date: Thursday, April 27, 2006
Time: 9:00-12:00
Duration: 180 minutes
Alert Type: Immediate
Context: Do not disturb
Created by: [Michael Hsu](#)
Email: mhsu@nps.edu
Cellphone Number: 8018152413
IM provider: AOL
IM screen-name: RunnerXBest
Alert Notification Method: email
Updated: Wednesday, April 26, 2006 15:21
Participants: [Michael Hsu](#)

[Printer Friendly](#)

[Edit entry](#)
[Delete entry](#)
[Copy entry](#)
[Email all participants](#)

Month: Week: Year:

IMAS v0.2 (1 Mar 2006)
[About Us](#) | [Site Map](#) | [Privacy Policy](#) | [Contact Us](#) | ©2006 Naval Postgraduate School

Figure 9. Do Not Disturb Context

In order to satisfy the Mobile alerts need to be timely and just in time requirement, a program written in php-script which converts emails into SMS text called *test.php*, listed in Appendix B was written. *Test.php* provides a means for incoming emails to automatically be forwarded to a user's cell-phone in SMS text. When the page is executed, *test.php* checks for newly arrived emails in the users email inbox. If there are new messages in the inbox, the program displays the pertinent message header information as well as the message and forwards the newly arrived message to the user's cell-phone.

Currently, the IMAS simulates a user's email account as being the user's IMAS data-store. *Test.php* uses the

user's own SMS gateway such as T-mobile, Cingular, and Verizon to route message traffic from their email account to their cellular phone. At present, *test.php* is only configured to retrieve and forward emails from IMAP servers. Figure 10 illustrates the output of *test.php* when two newly arrived emails arrive to a user's inbox.

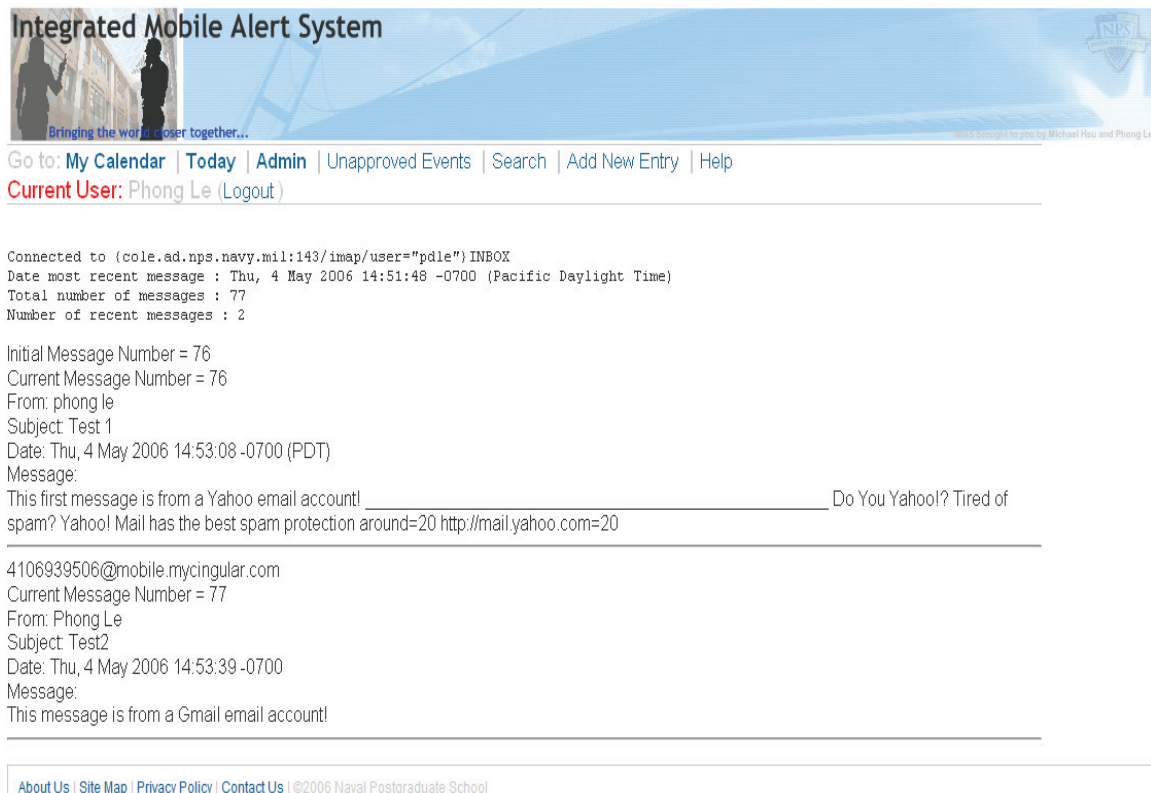


Figure 10. SMS Dissemination

C. DESIGN SETUP

This section describes the results of the physical and implementation phases. More specifically, a detailed discussion of the design setup is presented. This discussion includes details of the server and web-portal (including webcalendar) setup.

1. IMAS Server Setup

The IMAS is installed on a standard Dell PC equipped with an Intel Pentium 4 CPU clocking at 2.0GHz with 1.00 GB of RAM running Microsoft Windows Server 2003.

It employs the popular Database Management System (DBMS) called MySQL 5.0.18. During the project, all database administration decisions were made with MySQL Administrator. The DBMS performed well throughout its implementation and proved to be relatively easy to manage.

The logical choice in choosing database management applications to handle the administration of MySQL 5.0.18 over the internet is phpMyAdmin. It is the current database storage program where IMAS tables were created, dropped, and modified. IMAS also heavily employs the use of PHP-code scripting. The main benefit for using PHP lies in its ability to create and build dynamic web content, an important feature of the IMAS web portal.

2. IMAS Web-Portal

a. Design Considerations

There are two approaches used for developing a web portal: using static displays or dynamic content. In designing the IMAS web-portal, taking the static content approach is not a feasible design solution due to the requirement to call to the IMAS database. Working with massive amounts of data files would also render making changes a very tedious task in the static design approach.

The advantages of using dynamic content far outweigh the simplicity of the static design. These advantages include ease of modification, updates that apply to all pages on the fly, dynamically generated content, and data-source calls. The disadvantages of creating a dynamic content IMAS include a higher learning curve for the programming language, more in-depth knowledge of databases, and a thorough understanding of the server-client interfaces. However, due to the advantages that dynamic web-portals offer, IMAS is designed with the dynamic content approach.

b. Web Portal Implementation

The IMAS portal is designed to be easy to navigate, simple to look at, and easy for administrators to modify. Its format is based off of a generic Macromedia Dreamweaver 2004MX template called "Three-Column Left Nav." The template is located within Dreamweaver 2004MX under New Document/General/Page Designs (CSS). In addition to using this template, user created functions and cascading style sheets (CSS) were created to complete the IMAS portal design. A screenshot of the homepage, *index.php*, is displayed in Figure 11.

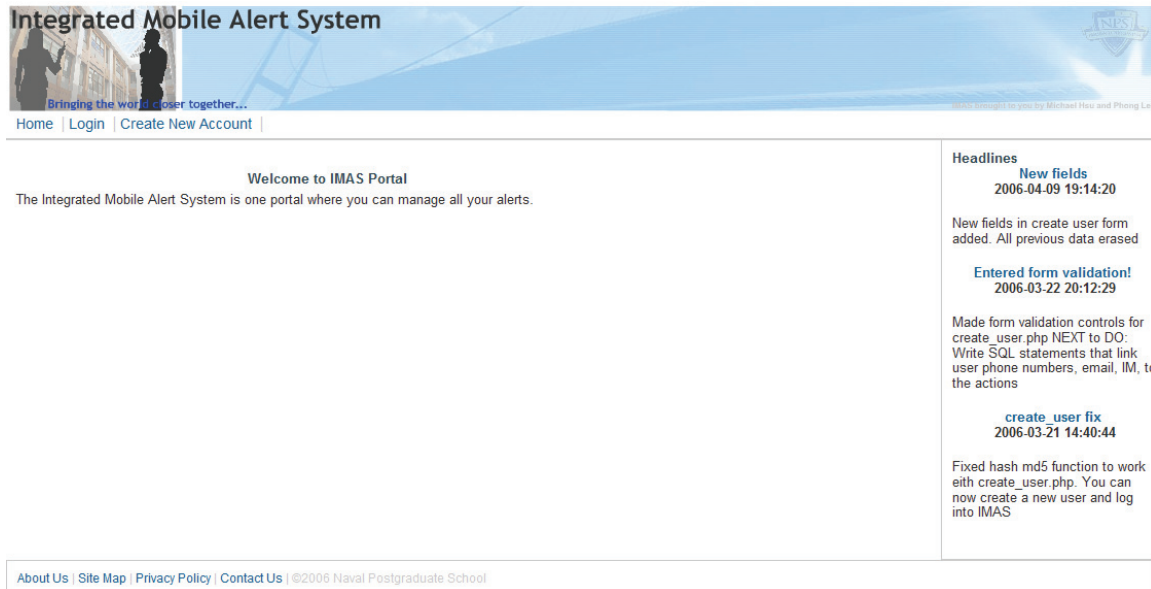


Figure 11. IMAS Homepage

There are three major components of the IMAS portal: Functions, CSS, and the main pages themselves.

The functions are contained in one file, *Class/index_Nav.php*, and includes all the database calls, footer information, navigation menus, and news items. Changes to any function will appear on every page that calls that particular function. The functions in *Class/index_NAV.php* reduce the need to modify each individual page.

Database calls are stored under the IMAS folder, *Connections/IMASxxxxDatabase.php*. The CSS contains page style parameters such as border-width, colors, font-style, size, and miscellaneous layout parameters. It also generates changes in all of the main pages that call to the style sheet.

Each webpage contains the page layout, specific style sheet used, function calls, and optional static

content if required. The main body, aka the dynamic content, is retrieved from the database thus rendering each page as a shell with no content by itself. In other words, the contents will not appear if the database does not exist. Each webpage is essentially a copy of a master webpage format that's slightly modified in order to include the function calls needed.

The IMAS portal utilizes the server-client architecture. It is designed to be modified with relative ease. All footer, header, navigational, and sidebar changes are made within the functions. Display changes such as color, border, text size, etc. are made in the CSS.

3. WebCalendar

An open-source PHP-based calendar application called WebCalendar is integrated with the IMAS web portal in order to satisfy the user profile interface requirement. WebCalendar is an open-source online calendar/scheduling system originally developed by Craig Knudsen, a developer at k5n solutions (www.k5n.us). The application can be run with several DBMS's including MySQL, PostgreSQL, Oracle, DB2, and ODBC. Some of its features include the following:

- A schedule management system for a single person.
- A schedule management system for a group of people, allowing one or more assistants to manage the calendar of another user.
- An events schedule that anyone can view, allowing visitors to submit new events.

- A calendar server that can be viewed with iCal-compliant calendar applications like [Mozilla Sunbird](#), [Apple iCal](#) or [GNOME Evolution](#) or RSS-enabled applications like [Firefox](#), [Thunderbird](#), [RSSOwl](#), or [FeedDemon](#), or [BlogExpress](#) (<http://www.k5n.us/webcalendar.php>).

Besides the above stated applications that WebCalendar can serve as, the program provides a means for IMAS users to configure their daily schedules and events. More importantly, it allows the user to manage the manner in which he/she wishes to be contacted during specific events.

In order to transform WebCalendar to support IMAS, a series of design modifications were made. First, WebCalendar web-pages were modified in order to provide a user input/output interface for tracking alert contexts and types. The modified scripting pages include *login.php*, *edit_entry.php*, and *view_entry.php*. Next, IMAS-specific tables were created to provide database entry support to IMAS. The overall database design and its tables will be discussed in the following chapter. In the end-state, the integration of WebCalendar with IMAS proved to be a feasible solution for users to input and track the manner in which they wish to be alerted. Those notification methods include email, IM notification, cellular phone call, and SMS text. As previously stated however, the current IMAS is limited in its ability to only support SMS text alert outputs.

D. IMAS DATABASE STRUCTURE

The IMAS database model follows the rules and naming conventions of the Integrated Definition 1 Extended (IDEF1X) standard. IDEF1X is an extension of the E-R model developed to support the creation of both conceptual and internal schemas. Developed in the mid-1980s for the U.S. military, the model was named a national standard in 1993 (Kroenke 2004, 46). The main differences between the E-R Model and IDEF1X are the naming convention differences in describing relationships. The following explanations listed below provide a sufficient understanding of IDEF1X relationships:

- 1) A parent entity is the entity where the relation originates from. A diamond is placed next to the entity to indicate if it's optional.
- 2) A child entity is the entity where the relation from the parent goes to. A filled black circle is placed next to the child entity. Additionally, a "Z", "1", and "P" can also be placed next to the child to indicate a zero or one requirement, one requirement, or positive number requirement respectively.
- 3) Non-Identifying Connection Relationships are one-to-one (1:1) or one-to-many (1:N) relationships between two non-ID dependent entities. It is similar to the HAS-A relationship in the E-R

model and is represented by a dashed line with a default cardinality of 1:N

- 4) Identifying Connection Relationships are similar to Non-Identifying Connection Relationships. The only differences are that child entities are ID-dependent on the parent entity and the relationship is represented by a solid line.
- 5) Non-Specific Relationships are many-to-many (N:M) relationships. According to the standard rules of normalization, it is prudent to avoid using N:M relationships.
- 6) Categorization Relationships are a specialization of generalization/subtype relationships between a generic entity and category entity.

1. IMAS E-R Diagram

Figure 14 displays the IMAS E-R diagram complete with attributes and relationships. It is the result of the data model in the conceptual phase. The structure of the WebCalendar database model is not included in the diagram shown.

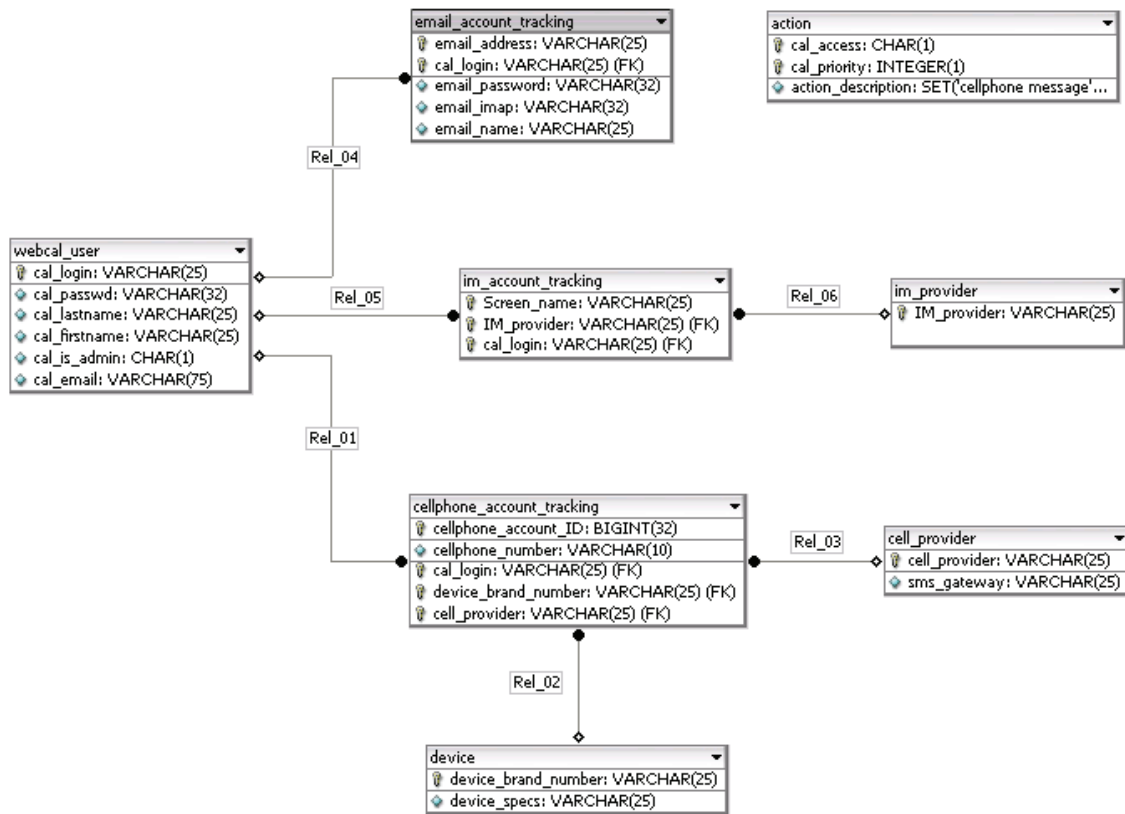


Figure 12. IMAS Database Model

2. IMAS Entities

During the conceptual design phase, various entity requirements were decided upon in order to construct a logical working system. The solution to the “*mobile alerts need to be seamless*” and “*alerts have to be context-sensitive*” requirements are addressed with the implementation of the eight entities listed in Figure 4.11.

This section will describe those IMAS-specific entities in great detail down to the attribute level.

The first table to consider, “webcal_user,” is the only table common to both the Webcalendar and the IMAS-

specific database. Webcal_user stores basic information about the user. More specifically, the fields cal_login, cal_password, cal_firstname, cal_lastname, cal_is_admin, and cal_email contain the users login name (primary key), login password, first name, last name, whether they have administrator privileges, and their email addresses respectively.

The second IMAS entity is "email_account_tracking." This table contains the following attributes: email_address (primary key), cal_login, email_name, email_imap and email_password. As the attribute titles imply, email_account_tracking tracks the user's email address, IMAS login name, email login name, IMAP server, and email password respectively.

The third entity, "cellphone_account_tracking," contains information about the users cell phone account. Cellphone_account_ID (primary key) is an auto-generated integer that assigns each user a unique account number. Cellphone_number stores the user's ten-digit cellular phone-number while cell_provider lists their cellular provider such as T-mobile, Cingular, Verizon, etc. Finally, cal_login and device_brand_number respectively captures the login name and cell-phone model.

The fourth table in the database is "device." This entity contains the attributes device_brand_number (primary key) and device_specs. Accordingly, the table stores the cell-phones model such as Motorola Razor and Nokia 6010 as well as its specifications respectively.

The next table, "cell_provider," is a simple entity with only one attribute also called cell_provider (primary key). This table lists the various cellular providers such

as T-mobile, Cingular, and Verizon. The primary purpose of the table is to allow users to select their provider from a drop-down menu when they create their profile in *create_user.php*.

The sixth table is labeled "im_account_tracking." It contains three fields. Screen_name stores the user's instant messaging (IM) service screen name. Next, IM_provider lists the IM service the user is registered with. The combination of both Screen_name and IM_provider composes the composite key. Finally, the third attribute, cal_login, stores the user's login name.

The next table, "im_provider," stores various IM providers. It is populated with only one attribute, IM_provider (primary key). Like the cell_provider table, im_provider exists to allow users to select an IM provider from a drop-down menu in *create_user.php*.

The eighth and final table, "action," is a key component of the IMAS database. It contains a composite key, cal_access and cal_priority which corresponds to event context and alert type respectively. Action also stores specific alert procedures labeled action_description. The various alert action IMAS takes is dependent on the specific event context and alert type that the user set *edit_entry.php*. Currently, the only alert type IMAS considers is "immediate" while the three contexts are "general", "meeting", and "do not disturb."

3. Relationships

The following relationships listed below are explained in reference to Figure 4.11. The placement of primary keys in relation to foreign keys is also displayed in the same Figure. Bear in mind that the database schema modeling tool used, DBDesigner 4, does not display dashed lines to indicate that whether or not a relationship is non-identifying.

The first relationship, Rel_01, is an identifying connection relationship originating from "webcal_user" to "cellphone_account_tracking." It states that a user has an optional cell-phone account. Cellphone_account_tracking is linked to its parent entity through the cal_login field and contains a cardinality of 1:1.

The second relationship is Rel_02. This is an identifying connection relationship originating from "device" to "cellphone_account_tracking." The relationship indicates that a cell-phone account, called cellphone_account_tracking, has a device. It is optional for a device to be connected to an account. Additionally, the cardinality between device and cellphone_account_tracking is 1:N.

Next, the third relationship, Rel_03, is an identifying connection relationship between "cellphone_account_tracking" and "cell_provider." Like Rel_02, Rel_03 states that a cell-phone account has a device. However, it is optional for the device to be connected to a cell-phone account. Again, the cardinality

between cell_provider and cellphone_account_tracking is 1:N.

The fourth relationship, Rel_04, is an identifying connection relationship originating from "webcal-user" to "email_account_tracking." A user can have an email account. Conversely, the email account does not have to have a user. The cardinality between webcal_user and email_account_tracking is 1:1.

The fifth relationship is Rel_05. Like Rel_04, Rel_05 is also a non-identifying connection relationship. It states that a "webcal_user" has an "im_account_tracking." The cardinality between the parent and child entities is 1:1. Additionally, the diamond by webcal_user indicates that im_account_tracking is optional.

The sixth and final relationship is labeled Rel_06. It is an identifying connection relationship which states that an "im_provider" has an optional "im_account_tracking." The cardinality between im_provider and im_account_tracking is 1:N.

4. WebCalendar Tables

WebCalendar employs an extensive and robust database consisting of twenty-three tables. This section will not explore each entity in great detail. Instead, a broad understanding of how each entity functions will be briefly described. The following definitions were taken from the WebCalendar 1.0 Database Documentation link online at "<http://webcalendar.sourceforge.net>."

- 1) Webcal_user: Defines a WebCalendar user.
- 2) Webcal_user_layers: Defines layers for a user.
- 3) Webcal_user_pref: Specifies user preferences.
- 4) Webcal_view: Allows a user to insert calendars of multiple users on one page.
- 5) Webcal_view_user: Specifies users in a view.
- 6) Webcal_entry: Defines a calendar event. Each event in the system has one entry in this table unless the event starts before midnight and ends after midnight. In that case a secondary event will be created with cal_ext_for_id set to the cal_id of the original entry.
- 7) Webcal_entry_ext_user: This table associates one or more external users (people who do not have a WebCalendar login) with an event by the event id. An event must still have at least one WebCalendar user associated with it.
- 8) Webcal_entry_log: Activity log for an event.
- 9) Webcal_entry_repeats: Defines repeating information about an event.
- 10) Webcal_entry_repeats_not: This table specifies which dates in a repeating event have either been deleted or replaced with a replacement event for that day.
- 11) Webcal_entry_user: This table associates one or more users with an event by the event id.
- 12) Webcal_group: Defines a group.
- 13) Webcal_group_user: Specifies users in a group.

- 14) Webcal_import: Used to track import data.
- 15) Webcal_import_data: Used to track import data.
- 16) Webcal_reminder_log: Keeps a history of when reminders get sent.
- 17) Webcal_report: Defines a custom report created by a user.
- 18) Webcal_report_template: Defines one of the templates used for a report.
- 19) Webcal_site_extras: This table holds data for site extra fields.
- 20) Webcal_config: System settings set by the admin interface in admin.php.
- 21) Webcal_categories: Defines user categories.
- 22) Webcal_asst: Defines the assistant/boss relationship.
- 23) Webcal_nonuser_cals: Defines non-user calendars.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The Integrated Mobile Alert System provides a mechanism to ensure mobile alerts arrive to their intended receivers. Users are not only guaranteed mobile alert reception but are given the capability to specify the manner they wish to receive alerts. IMAS improves C3 operations across a wide spectrum of users. Military commanders are able to control and communicate with their forces in tactical situations that hinder voice communications. Business professionals are kept abreast of the latest developments and decisions being made. Emergency responders are instantly notified of new crisis related information. However, to ensure that these alerts arrive at the right time in the right format requires a system capable of receiving various inputs and specifications and outputs correct alert formats.

This thesis provides the framework for designing and implementing the IMAS. It proves that DBMS' offer an efficient method for users to store mobile device account information as well as personal user data. A DBMS allows users and administrators to input, update, modify, and delete personal account information as they see fit. The open-source software used; MySQL and PHP, performed without any major issues during the implementation phase.

The database tables and their attributes are stored in MySQL and managed primarily using phpMyAdmin. MySQL Administrator and MySQL Query Browser were also used to

manage the MySQL database. These open-source solutions performed as well as commercial counterparts.

In order to determine how users could set the contexts of their daily schedules, a web-based calendar was embedded in the IMAS web-portal. This interface allows users to manage their daily schedules through an open-source web-based calendar called WebCalendar. By manipulating the scripts of WebCalendar, the program was successfully configured to meet the design requirements of IMAS. Users could specify the manner they wished to receive mobile alerts based on their various context profiles.

The creation of *test.php* integrated with the database validates the fact that mobile alerts can successfully be converted and disseminated between differing formats. In this case, *test.php* converts newly arrived emails into SMS texts and distributes the message to a receiver's cellular phone. In doing so, the capability demonstrates that it is possible to convert alerts from a data-store such as an email inbox into a mobile alert format the user wishes to receive. Future variations of IMAS should be able to process and disseminate more message format types with a more robust profiling algorithm.

B. RECOMMENDATIONS

This thesis provides a foundation for building a complete communication system that reduce information overload. There are many improvements and additional features that need to be implemented in order to fully

develop IMAS. Some of those recommendations are highlighted below:

- Modify the user input form to allow users to input multiple mobile device information and providers to be stored in the database.
- Develop more programming code to convert between other mobile alert formats. The current system only considers the conversion of email to SMS text.
- Modify the EASF code in order to allow continuous run-time execution. In other words, determine an optimal solution to run the EASF segment for 24 hrs a day and 7 days a week instead of limiting it to a few minutes.
- Implement the location, subscribed, and scheduled alert types in IMAS.
- Develop the ability to exchange information to desktop applications. By exchanging messages with desktop computers and static phone lines, greater connectivity coverage will be achieved.
- Develop an efficient manner to store personal mobile alerts in a data-store.
- Implement the emergency turn-on feature. Determine a method to turn on mobile devices during emergencies through a continuously polling and automatic wake-up feature.
- Determine a way for users to retrieve previous alert data from the data-store.

- At present, test.php only connects to IMAP email servers. Implement a way to retrieve and forward emails from POP3 servers.
- Develop a continuously polling feature in test.php to allow the system to run at set time intervals instead of only when the user executes the page.
- Email_account_tracking.email_imap in the database causes a connection error in test.php when white space is entered in the beginning of the field. For example, " cole.ad.nps.navy.mil" will cause a connection error in test.php vice "cole.ad.nps.navy.mil."

APPENDIX A. IMAS SYSTEM SETTINGS

A. MY.INI

```
# MySQL Server Instance Configuration File
# -----
-
# Generated by the MySQL Server Instance Configuration Wizard
#
#
# Installation Instructions
# -----
-
#
# On Linux you can copy this file to /etc/my.cnf to set global options,
# mysql-data-dir/my.cnf to set server-specific options
# (@localstatedir@ for this installation) or to
# ~/.my.cnf to set user-specific options.
#
# On Windows you should keep this file in the installation directory
# of your server (e.g. C:\Program Files\MySQL\MySQL Server 4.1). To
# make sure the server reads the config file use the startup option
# "--defaults-file".
#
# To run the server from the command line, execute this in a
# command line shell, e.g.
#   mysql --defaults-file="C:\Program Files\MySQL\MySQL Server
4.1\my.ini"
#
# To install the server as a Windows service manually, execute this in
a
# command line shell, e.g.
#   mysql --install MySQL41 --defaults-file="C:\Program
Files\MySQL\MySQL Server 4.1\my.ini"
#
# And then execute this in a command line shell to start the server,
e.g.
# net start MySQL41
#
#
# Guildlines for editing this file
# -----
-
#
# In this file, you can use all long options that the program supports.
# If you want to know the options a program supports, start the program
# with the "--help" option.
#
# More detailed information about the individual options can also be
# found in the manual.
#
#
# CLIENT SECTION
# -----
-
```

```

#
# The following options will be read by MySQL client applications.
# Note that only client applications shipped by MySQL are guaranteed
# to read this section. If you want your own MySQL client program to
# honor these values, you need to specify it as an option during the
# MySQL client library initialization.
#
[client]

port=3306

[mysql]

default-character-set=utf8


# SERVER SECTION
# -----
-
#
# The following options will be read by the MySQL Server. Make sure
# that
# you have installed the server correctly (see above) so it reads this
# file.
#
[mysqld]

# The TCP/IP Port the MySQL Server will listen on
port=3306


#Path to installation directory. All paths are usually resolved
relative to this.
basedir="C:/Program Files/MySQL/MySQL Server 5.0/"

#Path to the database root
datadir="C:/Program Files/MySQL/MySQL Server 5.0/Data/"

# The default character set that will be used when a new schema or
table is
# created and no character set is defined
default-character-set=utf8

# The default storage engine that will be used when create new tables
when
default-storage-engine=INNODB

# Set the SQL mode to strict
sql-
mode="STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION"

# The maximum amount of concurrent sessions the MySQL server will
# allow. One of these connections will be reserved for a user with
# SUPER privileges to allow the administrator to login even if the
# connection limit has been reached.
max_connections=160

```

```

# Query cache is used to cache SELECT results and later return them
# without actual executing the same query once again. Having the query
# cache enabled may result in significant speed improvements, if your
# have a lot of identical queries and rarely changing tables. See the
# "Qcache_lowmem_prunes" status variable to check if the current value
# is high enough for your load.
# Note: In case your tables change very often or if your queries are
# textually different every time, the query cache may result in a
# slowdown instead of a performance improvement.
query_cache_size=48M

# The number of open tables for all threads. Increasing this value
# increases the number of file descriptors that mysqld requires.
# Therefore you have to make sure to set the amount of open files
# allowed to at least 4096 in the variable "open-files-limit" in
# section [mysqld_safe]
table_cache=320

# Maximum size for internal (in-memory) temporary tables. If a table
# grows larger than this value, it is automatically converted to disk
# based table This limitation is for a single table. There can be many
# of them.
tmp_table_size=52M

# How many threads we should keep in a cache for reuse. When a client
# disconnects, the client's threads are put in the cache if there
# aren't
# more than thread_cache_size threads from before. This greatly
# reduces
# the amount of thread creations needed if you have a lot of new
# connections. (Normally this doesn't give a notable performance
# improvement if you have a good thread implementation.)
thread_cache_size=8

**** MyISAM Specific options

# The maximum size of the temporary file MySQL is allowed to use while
# recreating the index (during REPAIR, ALTER TABLE or LOAD DATA INFILE.
# If the file-size would be bigger than this, the index will be created
# through the key cache (which is slower).
myisam_max_sort_file_size=100G

# If the temporary file used for fast index creation would be bigger
# than using the key cache by the amount specified here, then prefer
# the
# key cache method. This is mainly used to force long character keys
# in
# large tables to use the slower key cache method to create the index.
myisam_max_extra_sort_file_size=100G

# If the temporary file used for fast index creation would be bigger
# than using the key cache by the amount specified here, then prefer
# the
# key cache method. This is mainly used to force long character keys
# in
# large tables to use the slower key cache method to create the index.

```

```

myisam_sort_buffer_size=80M

# Size of the Key Buffer, used to cache index blocks for MyISAM tables.
# Do not set it larger than 30% of your available memory, as some
memory
# is also required by the OS to cache rows. Even if you're not using
# MyISAM tables, you should still set it to 8-64M as it will also be
# used for internal temporary disk tables.
key_buffer_size=74M

# Size of the buffer used for doing full table scans of MyISAM tables.
# Allocated per thread, if a full scan is needed.
read_buffer_size=64K
read_rnd_buffer_size=256K

# This buffer is allocated when MySQL needs to rebuild the index in
# REPAIR, OPTIMIZE, ALTER table statements as well as in LOAD DATA
INFILE
# into an empty table. It is allocated per thread so be careful with
# large settings.
sort_buffer_size=256K

**** INNODB Specific options ****

# Use this option if you have a MySQL server with InnoDB support
enabled
# but you do not plan to use it. This will save memory and disk space
# and speed up some things.
#skip-innodb

# Additional memory pool that is used by InnoDB to store metadata
# information. If InnoDB requires more memory for this purpose it will
# start to allocate it from the OS. As this is fast enough on most
# recent operating systems, you normally do not need to change this
# value. SHOW INNODB STATUS will display the current amount used.
innodb_additional_mem_pool_size=3M

# If set to 1, InnoDB will flush (fsync) the transaction logs to the
# disk at each commit, which offers full ACID behavior. If you are
# willing to compromise this safety, and you are running small
# transactions, you may set this to 0 or 2 to reduce disk I/O to the
# logs. Value 0 means that the log is only written to the log file and
# the log file flushed to disk approximately once per second. Value 2
# means the log is written to the log file at each commit, but the log
# file is only flushed to disk approximately once per second.
innodb_flush_log_at_trx_commit=1

# The size of the buffer InnoDB uses for buffering log data. As soon as
# it is full, InnoDB will have to flush it to disk. As it is flushed
# once per second anyway, it does not make sense to have it very large
# (even with long transactions).
innodb_log_buffer_size=1478K

# InnoDB, unlike MyISAM, uses a buffer pool to cache both indexes and
# row data. The bigger you set this the less disk I/O is needed to

```

```

# access data in tables. On a dedicated database server you may set
this
# parameter up to 80% of the machine physical memory size. Do not set
it
# too large, though, because competition of the physical memory may
# cause paging in the operating system. Note that on 32bit systems you
# might be limited to 2-3.5G of user level memory per process, so do
not
# set it too high.
innodb_buffer_pool_size=143M

# Size of each log file in a log group. You should set the combined
size
# of log files to about 25%-100% of your buffer pool size to avoid
# unneeded buffer pool flush activity on log file overwrite. However,
# note that a larger logfile size will increase the time needed for the
# recovery process.
innodb_log_file_size=29M

# Number of threads allowed inside the InnoDB kernel. The optimal value
# depends highly on the application, hardware as well as the OS
# scheduler properties. A too high value may lead to thread thrashing.
innodb_thread_concurrency=8
#Use old password encryption method (needed for 4.0 and older clients).
#old-passwords
#Use old password encryption method (needed for 4.0 and older clients).
old-passwords

```

B. PHP.INI

```

[PHP]

;;;;;;;;;;
; WARNING ;
;;;;;;;;;;
; This is the default settings file for new PHP installations.
; By default, PHP installs itself with a configuration suitable for
; development purposes, and *NOT* for production purposes.
; For several security-oriented considerations that should be taken
; before going online with your site, please consult php.ini-
recommended
; and http://php.net/manual/en/security.php.

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;
; About this file ;
;;;;;;;;;;;;;;;;;;;;;;;;;
; This file controls many aspects of PHP's behavior. In order for PHP
to
; read it, it must be named 'php.ini'. PHP looks for it in the current
; working directory, in the path designated by the environment variable
; PHPRC, and in the path that was defined in compile time (in that
order).
; Under Windows, the compile-time path is the Windows directory. The
; path in which the php.ini file is looked for can be overridden using
; the -c argument in command line mode.
;

```



```

; The syntax of the file is extremely simple. Whitespace and Lines
; beginning with a semicolon are silently ignored (as you probably
guessed).
; Section headers (e.g. [Foo]) are also silently ignored, even though
; they might mean something in the future.
;
; Directives are specified using the following syntax:
; directive = value
; Directive names are *case sensitive* - foo=bar is different from
FOO=bar.
;
; The value can be a string, a number, a PHP constant (e.g. E_ALL or
M_PI), one
; of the INI constants (On, Off, True, False, Yes, No and None) or an
expression
; (e.g. E_ALL & ~E_NOTICE), or a quoted string ("foo").
;
; Expressions in the INI file are limited to bitwise operators and
parentheses:
; |           bitwise OR
; &           bitwise AND
; ~           bitwise NOT
; !           boolean NOT
;
; Boolean flags can be turned on using the values 1, On, True or Yes.
; They can be turned off using the values 0, Off, False or No.
;
; An empty string can be denoted by simply not writing anything after
the equal
; sign, or by using the None keyword:
;
; foo =           ; sets foo to an empty string
; foo = none      ; sets foo to an empty string
; foo = "none"    ; sets foo to the string 'none'
;
; If you use constants in your value, and these constants belong to a
; dynamically loaded extension (either a PHP extension or a Zend
extension),
; you may only use these constants *after* the line that loads the
extension.
;
; All the values in the php.ini-dist file correspond to the builtin
; defaults (that is, if no php.ini is used, or if you delete these
lines,
; the builtin defaults will be identical).

;;;;;;;;;;;;;;;;;;;;;;;;
; Language Options ;
;;;;;;;;;;;;;;;;;;;;;;;;

; Enable the PHP scripting language engine under Apache.
engine = On

; Allow the <? tag. Otherwise, only <?php and <script> tags are
recognized.

```

```

; NOTE: Using short tags should be avoided when developing applications
or
; libraries that are meant for redistribution, or deployment on PHP
; servers which are not under your control, because short tags may not
; be supported on the target server. For portable, redistributable
code,
; be sure not to use short tags.
short_open_tag = On

; Allow ASP-style tags.
asp_tags = Off

; The number of significant digits displayed in floating point numbers.
precision      = 12

; Enforce year 2000 compliance (will cause problems with non-compliant
browsers)
y2k_compliance = On

; Output buffering allows you to send header lines (including cookies)
even
; after you send body content, at the price of slowing PHP's output
layer a
; bit. You can enable output buffering during runtime by calling the
output
; buffering functions. You can also enable output buffering for all
files by
; setting this directive to On. If you wish to limit the size of the
buffer
; to a certain size - you can use a maximum number of bytes instead of
'On', as
; a value for this directive (e.g., output_buffering=4096).
output_buffering = Off

; You can redirect all of the output of your scripts to a function.
For
; example, if you set output_handler to "mb_output_handler", character
; encoding will be transparently converted to the specified encoding.
; Setting any output handler automatically turns on output buffering.
; Note: People who wrote portable scripts should not depend on this ini
; directive. Instead, explicitly set the output handler using
ob_start().
; Using this ini directive may cause problems unless you know
what script
; is doing.
; Note: You cannot use both "mb_output_handler" with "ob_iconv_handler"
; and you cannot use both "ob_gzhandler" and
"zlib.output_compression".
;output_handler =

; Transparent output compression using the zlib library
; Valid values for this option are 'off', 'on', or a specific buffer
size
; to be used for compression (default is 4KB)
; Note: Resulting chunk size may vary due to nature of compression. PHP
; outputs chunks that are few hundreds bytes each as a result of
; compression. If you prefer a larger chunk size for better

```

```

;      performance, enable output_buffering in addition.
; Note: You need to use zlib.output_handler instead of the standard
;      output_handler, or otherwise the output will be corrupted.
zlib.output_compression = Off

; You cannot specify additional output handlers if
zlib.output_compression
; is activated here. This setting does the same as output_handler but
in
; a different order.
;zlib.output_handler =

; Implicit flush tells PHP to tell the output layer to flush itself
; automatically after every output block. This is equivalent to
calling the
; PHP function flush() after each and every call to print() or echo()
and each
; and every HTML block. Turning this option on has serious performance
; implications and is generally recommended for debugging purposes
only.
implicit_flush = Off

; The unserialize callback function will be called (with the undefined
class'
; name as parameter), if the unserializer finds an undefined class
; which should be instantiated.
; A warning appears if the specified function is not defined, or if the
; function doesn't include/implement the missing class.
; So only set this entry, if you really want to implement such a
; callback-function.
unserialize_callback_func=

; Whether to enable the ability to force arguments to be passed by
reference
; at function call time. This method is deprecated and is likely to be
; unsupported in future versions of PHP/Zend. The encouraged method of
; specifying which arguments should be passed by reference is in the
function
; declaration. You're encouraged to try and turn this option Off and
make
; sure your scripts work properly with it in order to ensure they will
work
; with future versions of the language (you will receive a warning each
time
; you use this feature, and the argument will be passed by value
instead of by
; reference).
allow_call_time_pass_reference = On

; Safe Mode
;
safe_mode = Off

; By default, Safe Mode does a UID compare check when
; opening files. If you want to relax this to a GID compare,
; then turn on safe_mode_gid.
safe_mode_gid = Off

```

```

; When safe_mode is on, UID/GID checks are bypassed when
; including files from this directory and its subdirectories.
; (directory must also be in include_path or full path must
; be used when including)
safe_mode_include_dir =

; When safe_mode is on, only executables located in the
safe_mode_exec_dir
; will be allowed to be executed via the exec family of functions.
safe_mode_exec_dir =

; Setting certain environment variables may be a potential security
breach.
; This directive contains a comma-delimited list of prefixes. In Safe
Mode,
; the user may only alter environment variables whose names begin with the
; prefixes supplied here. By default, users will only be able to set
; environment variables that begin with PHP_ (e.g. PHP_FOO=BAR).
;
; Note: If this directive is empty, PHP will let the user modify ANY
; environment variable!
safe_mode_allowed_env_vars = PHP_

; This directive contains a comma-delimited list of environment
variables that
; the end user won't be able to change using putenv(). These variables
will be
; protected even if safe_mode_allowed_env_vars is set to allow to
change them.
safe_mode_protected_env_vars = LD_LIBRARY_PATH

; open_basedir, if set, limits all file operations to the defined
directory
; and below. This directive makes most sense if used in a per-
directory
; or per-virtualhost web server configuration file. This directive is
; *NOT* affected by whether Safe Mode is turned On or Off.
;open_basedir =

; This directive allows you to disable certain functions for security
reasons.
; It receives a comma-delimited list of function names. This directive
is
; *NOT* affected by whether Safe Mode is turned On or Off.
disable_functions =

; This directive allows you to disable certain classes for security
reasons.
; It receives a comma-delimited list of class names. This directive is
; *NOT* affected by whether Safe Mode is turned On or Off.
disable_classes =

; Colors for Syntax Highlighting mode. Anything that's acceptable in
; <font color="??????"> would work.
;highlight.string = #DD0000

```

```

;highlight.comment = #FF9900
;highlight.keyword = #007700
;highlight.bg      = #FFFFFF
;highlight.default = #0000BB
;highlight.html    = #000000

;
; Misc
;
; Decides whether PHP may expose the fact that it is installed on the
server
; (e.g. by adding its signature to the Web server header).  It is no
security
; threat in any way, but it makes it possible to determine whether you
use PHP
; on your server or not.
expose_php = On

;;;;;;;;;;;;;;;;;;;;;;;;
; Resource Limits ;
;;;;;;;;;;;;;;;;;;;;;;;;

max_execution_time = 30      ; Maximum execution time of each script, in
seconds
max_input_time = 60         ; Maximum amount of time each script may spend
parsing request data
memory_limit = 8M           ; Maximum amount of memory a script may consume
(8MB)

;;;;;;;;;;;;;;;;;;;;;;;;
; Error handling and logging ;
;;;;;;;;;;;;;;;;;;;;;;;;

; error_reporting is a bit-field.  Or each number up to get desired
error
; reporting level
; E_ALL           - All errors and warnings
; E_ERROR         - fatal run-time errors
; E_WARNING       - run-time warnings (non-fatal errors)
; E_PARSE        - compile-time parse errors
; E_NOTICE       - run-time notices (these are warnings which often
result
;                  from a bug in your code, but it's possible that
it was
;                  intentional (e.g., using an uninitialized
variable and
;                  relying on the fact it's automatically
initialized to an
;                  empty string)
; E_CORE_ERROR    - fatal errors that occur during PHP's initial
startup
; E_CORE_WARNING  - warnings (non-fatal errors) that occur during
PHP's
;                  initial startup
; E_COMPILE_ERROR - fatal compile-time errors

```

```

; E_COMPILE_WARNING - compile-time warnings (non-fatal errors)
; E_USER_ERROR      - user-generated error message
; E_USER_WARNING    - user-generated warning message
; E_USER_NOTICE     - user-generated notice message
;
; Examples:
;
;   - Show all errors, except for notices
;
;error_reporting = E_ALL & ~E_NOTICE
;
;   - Show only errors
;
;error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR
;
;   - Show all errors except for notices
;
error_reporting = E_ALL & ~E_NOTICE

; Print out errors (as a part of the output). For production web
sites,
; you're strongly encouraged to turn this feature off, and use error
logging
; instead (see below). Keeping display_errors enabled on a production
web site
; may reveal security information to end users, such as file paths on
your Web
; server, your database schema or other information.
display_errors = On

; Even when display_errors is on, errors that occur during PHP's
startup
; sequence are not displayed. It's strongly recommended to keep
; display_startup_errors off, except for when debugging.
display_startup_errors = Off

; Log errors into a log file (server-specific log, stderr, or error_log
(below))
; As stated above, you're strongly advised to use error logging in
place of
; error displaying on production web sites.
log_errors = Off

; Set maximum length of log_errors. In error_log information about the
source is
; added. The default is 1024 and 0 allows to not apply any maximum
length at all.
log_errors_max_len = 1024

; Do not log repeated messages. Repeated errors must occur in same file
on same
; line until ignore_repeated_source is set true.
ignore_repeated_errors = Off

; Ignore source of message when ignoring repeated messages. When this
setting

```

```

; is On you will not log errors with repeated messages from different
files or
; sourcelines.
ignore_repeated_source = Off

; If this parameter is set to Off, then memory leaks will not be shown
(on
; stdout or in the log). This has only effect in a debug compile, and
if
; error reporting includes E_WARNING in the allowed list
report_memleaks = On

; Store the last error/warning message in $php_errormsg (boolean).
track_errors = Off

; Disable the inclusion of HTML tags in error messages.
;html_errors = Off

; If html_errors is set On PHP produces clickable error messages that
direct
; to a page describing the error or function causing the error in
detail.
; You can download a copy of the PHP manual from
http://www.php.net/docs.php
; and change docref_root to the base URL of your local copy including
the
; leading '/'. You must also specify the file extension being used
including
; the dot.
;docref_root = "/phpmanual/"
;docref_ext = .html

; String to output before an error message.
;error_prepend_string = "<font color=ff0000>"

; String to output after an error message.
;error_append_string = "</font>"

; Log errors to specified file.
;error_log = filename

; Log errors to syslog (Event Log on NT, not valid in Windows 95).
;error_log = syslog

;;;;;;;;;;;;;;;;;;;;;;;;;
; Data Handling ;
;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Note - track_vars is ALWAYS enabled as of PHP 4.0.3

; The separator used in PHP generated URLs to separate arguments.
; Default is "&".
;arg_separator.output = "&"

; List of separator(s) used by PHP to parse input URLs into variables.
; Default is "&".

```

```

; NOTE: Every character in this directive is considered as separator!
;arg_separator.input = "&"

; This directive describes the order in which PHP registers GET, POST,
Cookie,
; Environment and Built-in variables (G, P, C, E & S respectively,
often
; referred to as EGPCS or GPC). Registration is done from left to
right, newer
; values override older values.
variables_order = "EGPCS"

; Whether or not to register the EGPCS variables as global variables.
You may
; want to turn this off if you don't want to clutter your scripts'
global scope
; with user data. This makes most sense when coupled with track_vars -
in which
; case you can access all of the GPC variables through the
$_HTTP_*_VARS[],
; variables.
;
; You should do your best to write your scripts so that they do not
require
; register_globals to be on; Using form variables as globals can
easily lead
; to possible security problems, if the code is not very well thought
of.
register_globals = On

; This directive tells PHP whether to declare the argv&argc variables
(that
; would contain the GET information). If you don't use these
variables, you
; should turn it off for increased performance.
register_argc_argv = On

; Maximum size of POST data that PHP will accept.
post_max_size = 8M

; This directive is deprecated. Use variables_order instead.
gpc_order = "GPC"

; Magic quotes
;

; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = on

; Magic quotes for runtime-generated data, e.g. data from SQL, from
exec(), etc.
magic_quotes_runtime = Off

; Use Sybase-style magic quotes (escape ' with '' instead of \').
magic_quotes_sybase = Off

; Automatically add files before or after any PHP document.

```



```

auto_prepend_file =
auto_append_file =

; As of 4.0b4, PHP always outputs a character encoding by default in
; the Content-type: header. To disable sending of the charset, simply
; set it to be empty.
;
; PHP's built-in default is text/html
default_mimetype = "text/html"
default_charset = "iso-8859-1"

; Always populate the $HTTP_RAW_POST_DATA variable.
always_populate_raw_post_data = On

;;;;;;;;;;;;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;;;;;;;;;;;;

; UNIX: "/path1:/path2"
include_path = ".: "
;
; Windows: "\\path1;\\path2"

;***** Added by go-pear
include_path=".;C:\php\includes\pear"
;*****

; The root of the PHP pages, used only if nonempty.
; if PHP was not compiled with FORCE_REDIRECT, you SHOULD set doc_root
; if you are running php as a CGI under any web server (other than IIS)
; see documentation for security issues. The alternate is to use the
; cgi.force_redirect configuration below
; doc_root =

; The directory under which PHP opens the script using ~/username used
only
; if nonempty.
user_dir =

; Directory in which the loadable extensions (modules) reside.
extension_dir = "C:\php\php\extensions"

; Whether or not to enable the dl() function. The dl() function does
NOT work
; properly in multithreaded servers, such as IIS or Zeus, and is
automatically
; disabled on them.
enable_dl = On

; cgi.force_redirect is necessary to provide security running PHP as a
CGI under
; most web servers. Left undefined, PHP turns this on by default. You
can
; turn it off here AT YOUR OWN RISK

```

```

; **You CAN safely turn this off for IIS, in fact, you MUST.**
cgi.force_redirect = 0

; if cgi.force_redirect is turned on, and you are not running under
Apache or Netscape
; (iPlanet) web servers, you MAY need to set an environment variable
name that PHP
; will look for to know it is OK to continue execution. Setting this
variable MAY
; cause security issues, KNOW WHAT YOU ARE DOING FIRST.
; cgi.redirect_status_env = 1;

; FastCGI under IIS (on WINNT based OS) supports the ability to
impersonate
; security tokens of the calling client. This allows IIS to define the
; security context that the request runs under. mod_fastcgi under
Apache
; does not currently support this feature (03/17/2002)
; Set to 1 if running under IIS. Default is zero.
fastcgi.impersonate = 1;

; cgi.rfc2616_headers configuration option tells PHP what type of
headers to
; use when sending HTTP response code. If it's set 0 PHP sends Status:
header that
; is supported by Apache. When this option is set to 1 PHP will send
; RFC2616 compliant header.
; Set to 1 if running under IIS. Default is zero.
cgi.rfc2616_headers = 0

;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;

; Whether to allow HTTP file uploads.
file_uploads = On

; Temporary directory for HTTP uploaded files (will use system default
if not
; specified).
;upload_tmp_dir = "C:\php\uploads"

; Maximum allowed size for uploaded files.
upload_max_filesize = 2M

;;;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as
files.
allow_url_fopen = On

; Define the anonymous ftp password (your email address)
;from="john@doe.com"

```

```

; Define the User-Agent string
; user_agent="PHP"

; Default timeout for socket based streams (seconds)
default_socket_timeout = 60

; If your scripts have to deal with files from Macintosh systems,
; or you are running on a Mac and need to deal with files from
; unix or win32 systems, setting this flag will cause PHP to
; automatically detect the EOL character in those files so that
; fgets() and file() will work regardless of the source of the file.
; auto_detect_line_endings = Off

;;;;;;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;;;;;;
;
; If you wish to have an ;ion loaded automatically, use the following
; syntax:
;
;   extension=module.name.extension
;
; For example, on Windows:
;
;   extension=mysql.dll
;
; ... or under UNIX:
;
;   extension=mysql.so
;
; Note that it should be the name of the module only; no directory
information
; needs to go here. Specify the location of the extension with the
; extension_dir directive above.

;Windows Extensions
;Note that MySQL and ODBC support is now built in, so no dll is needed
for it.
;

extension=php_sqlite.dll
;extension=php_mssql.dll
;extension=php_oci8.dll
;extension=php_pgsql.dll
;extension=php_adodb.dll

;extension=php_imap.dll
extension=php_curl.dll
extension=php_gd2.dll

;extension=php_bz2.dll
;extension=php_cpdf.dll
;extension=php_crack.dll
;extension=php_db.dll

```

```

;extension=php_dba.dll
;extension=php_dbase.dll
;extension=php_dbx.dll
;extension=php_domxml.dll
;extension=php_exif.dll
;extension=php_fdf.dll
;extension=php_filepro.dll
;extension=php_gettext.dll
;extension=php_hyperwave.dll
;extension=php_iconv.dll
;extension=php_ifx.dll
extension=php_iisfunc.dll
;extension=php_interbase.dll
extension=php_imap.dll
extension=php_java.dll
;extension=php_ldap.dll
extension=php_mbstring.dll
;extension=php_mcrypt.dll
extension=php_mhash.dll
;extension=php_mime_magic.dll
;extension=php_ming.dll
;extension=php_mysql.dll
;extension=php_openssl.dll
;extension=php_oracle.dll
;extension=php_pdf.dll
;extension=php_printer.dll
;extension=php_shmop.dll
;extension=php_snmp.dll
;extension=php_sockets.dll
;extension=php_sybase_ct.dll
;extension=php_w32api.dll
;extension=php_xmlrpc.dll
;extension=php_xslt.dll
;extension=php_yaz.dll
;extension=php_zip.dll

```

```

;;;;;;;;;;;;;;
; Module Settings ;
;;;;;;;;;;;;;;

```

```

[Syslog]
; Whether or not to define the various syslog variables (e.g. $LOG_PID,
; $LOG_CRON, etc.). Turning it off is a good idea performance-wise.
In
; runtime, you can define these variables by calling
define_syslog_variables().
define_syslog_variables = Off

```

```

[mail function]
; For Win32 only.
SMTP = cole.ad.nps.navy.mil

```

```

; For Win32 only.
sendmail_from = imas@nps.edu

```

```

; For Unix only.  You may supply arguments as well (default: "sendmail
-t -i").
;sendmail_path =

[Java]
java.class.path = .\php_java.jar
java.home = c:\j2sdk1.4.2_03
java.library = c:\j2sdk1.4.2_03\jre\bin\server\jvm.dll
java.library.path = .\

[SQL]
sql.safe_mode = Off

[ODBC]
;odbc.default_db      = Not yet implemented
;odbc.default_user    = Not yet implemented
;odbc.default_pw      = Not yet implemented

; Allow or prevent persistent links.
odbc.allow_persistent = On

; Check that a connection is still valid before reuse.
odbc.check_persistent = On

; Maximum number of persistent links.  -1 means no limit.
odbc.max_persistent = -1

; Maximum number of links (persistent + non-persistent).  -1 means no
limit.
odbc.max_links = -1

; Handling of LONG fields.  Returns number of bytes to variables.  0
means
; passthru.
odbc.defaultlrl = 4096

; Handling of binary data.  0 means passthru, 1 return as is, 2 convert
to char.
; See the documentation on odbc_binmode and odbc_longreadlen for an
explanation
; of uodbc.defaultlrl and uodbc.defaultbinmode
odbc.defaultbinmode = 1

[MySQL]
; Allow or prevent persistent links.
mysql.allow_persistent = On

; Maximum number of persistent links.  -1 means no limit.
mysql.max_persistent = -1

; Maximum number of links (persistent + non-persistent).  -1 means no
limit.
mysql.max_links = -1

; Default port number for mysql_connect().  If unset, mysql_connect()
will use
; the $MYSQL_TCP_PORT or the mysql-tcp entry in /etc/services or the

```

```

; compile-time value defined MYSQL_PORT (in that order).  Win32 will
only look
; at MYSQL_PORT.
mysql.default_port =

; Default socket name for local MySQL connects.  If empty, uses the
built-in
; MySQL defaults.
mysql.default_socket =

; Default host for mysql_connect() (doesn't apply in safe mode).
mysql.default_host =

; Default user for mysql_connect() (doesn't apply in safe mode).
mysql.default_user =

; Default password for mysql_connect() (doesn't apply in safe mode).
; Note that this is generally a *bad* idea to store passwords in this
file.
; *Any* user with PHP access can run 'echo
get_cfg_var("mysql.default_password")
; and reveal this password!  And of course, any users with read access
to this
; file will be able to reveal the password as well.
mysql.default_password =

; Maximum time (in secondes) for connect timeout. -1 means no limit
mysql.connect_timeout = -1

; Trace mode. When trace_mode is active (=On), warnings for table/index
scans and
; SQL-Errors will be displayed.
mysql.trace_mode = Off

[MySQL]
; Allow or prevent persistent links.
mysql.allow_persistent = On

; Maximum number of persistent links. -1 means no limit.
mysql.max_persistent = -1

; Maximum number of links (persistent+non persistent). -1 means no
limit.
mysql.max_links = -1

[PostgreSQL]
; Allow or prevent persistent links.
pgsql.allow_persistent = On

; Detect broken persistent links always with pg_pconnect(). Need a
little overhead.
pgsql.auto_reset_persistent = Off

; Maximum number of persistent links. -1 means no limit.
pgsql.max_persistent = -1

```

```

; Maximum number of links (persistent+non persistent).  -1 means no
limit.
pgsql.max_links = -1

; Ignore PostgreSQL backends Notice message or not.
pgsql.ignore_notice = 0

; Log PostgreSQL backends Noitce message or not.
; Unless pgsql.ignore_notice=0, module cannot log notice message.
pgsql.log_notice = 0

[Sybase]
; Allow or prevent persistent links.
sybase.allow_persistent = On

; Maximum number of persistent links.  -1 means no limit.
sybase.max_persistent = -1

; Maximum number of links (persistent + non-persistent).  -1 means no
limit.
sybase.max_links = -1

;sybase.interface_file = "/usr/sybase/interfaces"

; Minimum error severity to display.
sybase.min_error_severity = 10

; Minimum message severity to display.
sybase.min_message_severity = 10

; Compatability mode with old versions of PHP 3.0.
; If on, this will cause PHP to automatically assign types to results
according
; to their Sybase type, instead of treating them all as strings.  This
; compatability mode will probably not stay around forever, so try
applying
; whatever necessary changes to your code, and turn it off.
sybase.compatability_mode = Off

[Sybase-CT]
; Allow or prevent persistent links.
sybct.allow_persistent = On

; Maximum number of persistent links.  -1 means no limit.
sybct.max_persistent = -1

; Maximum number of links (persistent + non-persistent).  -1 means no
limit.
sybct.max_links = -1

; Minimum server message severity to display.
sybct.min_server_severity = 10

; Minimum client message severity to display.
sybct.min_client_severity = 10

[dbx]

```

```

; returned column names can be converted for compatibility reasons
; possible values for dbx.colnames_case are
; "unchanged" (default, if not set)
; "lowercase"
; "uppercase"
; the recommended default is either upper- or lowercase, but
; unchanged is currently set for backwards compatibility
dbx.colnames_case = "unchanged"

[bcmath]
; Number of decimal digits for all bcmath functions.
bcmath.scale = 0

[browscap]
;browscap = extra/browscap.ini

[Informix]
; Default host for ifx_connect() (doesn't apply in safe mode).
ifx.default_host =

; Default user for ifx_connect() (doesn't apply in safe mode).
ifx.default_user =

; Default password for ifx_connect() (doesn't apply in safe mode).
ifx.default_password =

; Allow or prevent persistent links.
ifx.allow_persistent = On

; Maximum number of persistent links. -1 means no limit.
ifx.max_persistent = -1

; Maximum number of links (persistent + non-persistent). -1 means no
limit.
ifx.max_links = -1

; If on, select statements return the contents of a text blob instead
of its id.
ifx.textasvarchar = 0

; If on, select statements return the contents of a byte blob instead
of its id.
ifx.byteasvarchar = 0

; Trailing blanks are stripped from fixed-length char columns. May
help the
; life of Informix SE users.
ifx.charasvarchar = 0

; If on, the contents of text and byte blobs are dumped to a file
instead of
; keeping them in memory.
ifx.blobinfile = 0

; NULL's are returned as empty strings, unless this is set to 1. In
that case,
; NULL's are returned as string 'NULL'.

```



```

ifx.nullformat = 0

[Session]
; Handler used to store/retrieve data.
session.save_handler = files

; Argument passed to save_handler. In the case of files, this is the
path
; where data files are stored. Note: Windows users have to change this
; variable in order to use PHP's session functions.
; As of PHP 4.0.1, you can define the path as:
;     session.save_path = "N;/path"
; where N is an integer. Instead of storing all the session files in
; /path, what this will do is use subdirectories N-levels deep, and
; store the session data in those directories. This is useful if you
; or your OS have problems with lots of files in one directory, and is
; a more efficient layout for servers that handle lots of sessions.
; NOTE 1: PHP will not create this directory structure automatically.
;         You can use the script in the ext/session dir for that
purpose.
; NOTE 2: See the section on garbage collection below if you choose to
;         use subdirectories for session storage
session.save_path = "C:\php\sessions"

; Whether to use cookies.
session.use_cookies = 1

; This option enables administrators to make their users invulnerable
to
; attacks which involve passing session ids in URLs; defaults to 0.
; session.use_only_cookies = 1

; Name of the session (used as cookie name).
session.name = PHPSESSID

; Initialize session on request startup.
session.auto_start = 0

; Lifetime in seconds of cookie or, if 0, until browser is restarted.
session.cookie_lifetime = 0

; The path for which the cookie is valid.
session.cookie_path = "C:\php\sessions"

; The domain for which the cookie is valid.
session.cookie_domain =

; Handler used to serialize data. php is the standard serializer of
PHP.
session.serialize_handler = php

; Define the probability that the 'garbage collection' process is
started
; on every session initialization.
; The probability is calculated by using gc_probability/gc_divisor,
; e.g. 1/100 means there is a 1 percent chance that the GC process
starts

```

```

; on each request.

session.gc_probability = 1
session.gc_divisor     = 100

; After this number of seconds, stored data will be seen as 'garbage'
and
; cleaned up by the garbage collection process.
session.gc_maxlifetime = 1440

; NOTE: If you are using the subdirectory option for storing session
files
;       (see session.save_path above), then garbage collection does
*not*
;       happen automatically. You will need to do your own garbage
;       collection through a shell script, cron entry, or some other
method.
;       For example, the following script would is the equivalent of
;       setting session.gc_maxlifetime to 1440 (1440 seconds = 24
minutes):
;       cd /path/to/sessions; find -cmin +24 | xargs rm

; PHP 4.2 and less have an undocumented feature/bug that allows you to
; to initialize a session variable in the global scope, albeit
register_globals
; is disabled. PHP 4.3 and later will warn you, if this feature is
used.
; You can disable the feature and the warning seperately. At this time,
; the warning is only displayed, if bug_compat_42 is enabled.

session.bug_compat_42 = 0
session.bug_compat_warn = 0

; Check HTTP Referer to invalidate externally stored URLs containing
ids.
; HTTP_REFERER has to contain this substring for the session to be
; considered as valid.
session.referer_check =

; How many bytes to read from the file.
session.entropy_length = 0

; Specified here to create the session id.
session.entropy_file =

;session.entropy_length = 16

;session.entropy_file = /dev/urandom

; Set to {nocache,private,public,} to determine HTTP caching aspects
; or leave this empty to avoid sending anti-caching headers.
session.cache_limiter = nocache

; Document expires after n minutes.
session.cache_expire = 180

; trans sid support is disabled by default.

```

```

; Use of trans sid may risk your users security.
; Use this option with caution.
; - User may send URL contains active session ID
;   to other person via. email/irc/etc.
; - URL that contains active session ID may be stored
;   in publically accessible computer.
; - User may access your site with the same session ID
;   always using URL stored in browser's history or bookmarks.
session.use_trans_sid = 0

; The URL rewriter will look for URLs in a defined set of HTML tags.
; form/fieldset are special; if you include them here, the rewriter
will
; add a hidden <input> field with the info which is otherwise appended
; to URLs. If you want XHTML conformity, remove the form entry.
; Note that all valid entries require a "=", even if no value follows.
url_rewriter.tags                                     =
"a=href,area=href,frame=src,input=src,form=,fieldset="

[MSSQL]
; Allow or prevent persistent links.
mssql.allow_persistent = On

; Maximum number of persistent links. -1 means no limit.
mssql.max_persistent = -1

; Maximum number of links (persistent+non persistent). -1 means no
limit.
mssql.max_links = -1

; Minimum error severity to display.
mssql.min_error_severity = 10

; Minimum message severity to display.
mssql.min_message_severity = 10

; Compatability mode with old versions of PHP 3.0.
mssql.compatability_mode = Off

; Valid range 0 - 2147483647. Default = 4096.
;mssql.textlimit = 4096

; Valid range 0 - 2147483647. Default = 4096.
;mssql.textsize = 4096

; Limits the number of records in each batch. 0 = all records in one
batch.
;mssql.batchsize = 0

; Use NT authentication when connecting to the server
mssql.secure_connection = Off

; Specify max number of processes. Default = 25
;mssql.max_procs = 25

[Assertion]
; Assert(expr); active by default.

```

```

;assert.active = On

; Issue a PHP warning for each failed assertion.
;assert.warning = On

; Don't bail out by default.
;assert.bail = Off

; User-function to be called if an assertion fails.
;assert.callback = 0

; Eval the expression with current error_reporting(). Set to true if
you want
; error_reporting(0) around the eval().
;assert.quiet_eval = 0

[Ingres II]
; Allow or prevent persistent links.
ingres.allow_persistent = On

; Maximum number of persistent links. -1 means no limit.
ingres.max_persistent = -1

; Maximum number of links, including persistents. -1 means no limit.
ingres.max_links = -1

; Default database (format: [node_id::]dbname[/srv_class]).
ingres.default_database =

; Default user.
ingres.default_user =

; Default password.
ingres.default_password =

[Verisign Payflow Pro]
; Default Payflow Pro server.
pfpro.defaulthost = "test-payflow.verisign.com"

; Default port to connect to.
pfpro.defaultport = 443

; Default timeout in seconds.
pfpro.defaulttimeout = 30

; Default proxy IP address (if required).
pfpro.proxyaddress =

; Default proxy port.
pfpro.proxyport =

; Default proxy logon.
pfpro.proxylogon =

; Default proxy password.
pfpro.proxypassword =

```

```

[Socket]
; Use the system read() function instead of the php_read() wrapper.
sockets.use_system_read = On

[com]
; path to a file containing GUIDs, IIDs or filenames of files with
TypeLibs
;com.typelib_file =
; allow Distributed-COM calls
;com.allow_dcom = true
; autoregister constants of a components typelib on com_load()
;com.autoregister_typelib = true
; register constants casesensitive
;com.autoregister_casesensitive = false
; show warnings on duplicate constat registrations
;com.autoregister_verbose = true

[Printer]
;printer.default_printer = ""

[mbstring]
; language for internal character representation.
;mbstring.language = Japanese

; internal/script encoding.
; Some encoding cannot work as internal encoding.
; (e.g. SJIS, BIG5, ISO-2022-*)
;mbstring.internal_encoding = EUC-JP

; http input encoding.
;mbstring.http_input = auto

; http output encoding. mb_output_handler must be
; registered as output buffer to function
;mbstring.http_output = SJIS

; enable automatic encoding translation accoding to
; mbstring.internal_encoding setting. Input chars are
; converted to internal encoding by setting this to On.
; Note: Do _not_ use automatic encoding translation for
;      portable libs/applications.
;mbstring.encoding_translation = Off

; automatic encoding detection order.
; auto means
;mbstring.detect_order = auto

; substitute_character used when character cannot be converted
; one from another
;mbstring.substitute_character = none;

; overload(replace) single byte functions by mbstring functions.
; mail(), ereg(), etc are overloaded by mb_send_mail(), mb_ereg(),
; etc. Possible values are 0,1,2,4 or combination of them.
; For example, 7 for overload everything.
; 0: No overload
; 1: Overload mail() function

```

```

; 2: Overload str*() functions
; 4: Overload ereg*() functions
;mbstring.func_overload = 0

[FrontBase]
;fbsql.allow_persistent = On
;fbsql.autocommit = On
;fbsql.default_database =
;fbsql.default_database_password =
;fbsql.default_host =
;fbsql.default_password =
;fbsql.default_user = "_SYSTEM"
;fbsql.generate_warnings = Off
;fbsql.max_connections = 128
;fbsql.max_links = 128
;fbsql.max_persistent = -1
;fbsql.max_results = 128
;fbsql.batchSize = 1000

[Crack]
; Modify the setting below to match the directory location of the
cracklib
; dictionary files. Include the base filename, but not the file
extension.
; crack.default_dictionary = "c:\php\lib\cracklib_dict"

[exif]
; Exif UNICODE user comments are handled as UCS-2BE/UCS-2LE and JIS as
JIS.
; With mbstring support this will automatically be converted into the
encoding
; given by corresponding encode setting. When empty
mbstring.internal_encoding
; is used. For the decode settings you can distinguish between motorola
and
; intel byte order. A decode setting cannot be empty.
;exif.encode_unicode = ISO-8859-15
;exif.decode_unicode_motorola = UCS-2BE
;exif.decode_unicode_intel = UCS-2LE
;exif.encode_jis =
;exif.decode_jis_motorola = JIS
;exif.decode_jis_intel = JIS

; Local Variables:
; tab-width: 4
; End:

[mmcache]

zend_extension_ts="C:\php\turck-mmcache\mmcache.dll"

; Amount of shared memory in MB to allocate for cache
mmcache.shm_size="16"
mmcache.cache_dir="C:\php\turck-mmcache\temp"
mmcache.enable="1"
mmcache.optimizer="1"
mmcache.check_mtime="1"

```

```
mmcache.debug="0"  
mmcache.filter=""  
mmcache.shm_max="5120000"  
mmcache.shm_ttl="3600"  
mmcache.shm_prune_period="0"  
  
[Zend]  
zend_optimizer.optimization_level=15  
zend_extension_ts="C:\php\Zend\lib\ZendExtensionManager.dll"  
zend_extension_manager.optimizer_ts="C:\php\Zend\lib\Optimizer-2.5.7"
```

APPENDIX B. IMAS CODE

A. CREATE_USER.PHP

```
<?php
include "../Class/index_Nav.php";
include "includes/config.php";
include "includes/php-dbi.php";
include "includes/functions.php";
include "includes/$user_inc";
include "includes/connect.php";
session_start();

//Original Webcalendar CSS Style Sheet - commented out on February 28,
2006
/* include "includes/styles.php"; */

// Print custom header (since we do not call print_header function)
if ( ! empty ( $CUSTOM_SCRIPT ) && $CUSTOM_SCRIPT == 'Y' ) {
    $res = dbi_query (
        "SELECT cal_template_text FROM webcal_report_template " .
        "WHERE cal_template_type = 'S' and cal_report_id = 0" );
    if ( $res ) {
        if ( $row = dbi_fetch_row ( $res ) ) {
            echo $row[0];
        }
        dbi_free_result ( $res );
    }
}
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<!-- DW6 -->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<title>IMAS Portal</title>
<link rel="stylesheet" href="../CSS_Styles/Portal_Public.css"
type="text/css">
<script language="JavaScript" src="includes/gen_validatorv2.js"
type="text/javascript"></script>
</head>

<script type="text/javascript">
// error check login/password

function myOnLoad() {
    <?php if ( ! empty ( $plugins_enabled ) && ( $plugins_enabled ) ){ ?>
        if (self != top) {
            window.open("login.php","_top","");
            return;
        }
    }
}
```



```

<?php } ?>
document.login_form.login.focus();
<?php
    if ( ! empty ( $login ) ) echo
"document.login_form.login.select();"
    if ( ! empty ( $error ) ) {
        echo " alert ( \"$error\" );\n";
    }
?>
}

```

//Various form validation functions for create_user.php - March 22, 2006

```

function emailvalidation(entered, alertbox)
{
with (entered)
{
apos=value.indexOf("@");
dotpos=value.lastIndexOf(".");
lastpos=value.length-1;
if (apos<1 || dotpos-apos<2 || lastpos-dotpos>3 || lastpos-dotpos<2)
{if (alertbox) {alert(alertbox);} return false;}
else {return true;}
}
}

```

```

function digitvalidation(entered, min, max, alertbox, datatype)
{
with (entered)
{
checkvalue=parseFloat(value);
if (datatype)
{smalldatatype=datatype.toLowerCase();
if (smalldatatype.charAt(0)=="i")
{checkvalue=parseInt(value);          if (value.indexOf(".")!=-1)
{checkvalue=checkvalue+1}};
}
if ((parseFloat(min)==min && value.length<min) || (parseFloat(max)==max
&& value.length>max) || value!=checkvalue)
{if (alertbox!="") {alert(alertbox);} return false;}
else {return true;}
}
}

```

```

function emptyvalidation(entered, alertbox)
{
with (entered)
{
if (value==null || value=="")
{if (alertbox!="") {alert(alertbox);} return false;}
else {return true;}
}
}

```

</script>

```

<body onload="myOnLoad();">
<div id="masthead">
  <h1 id="siteName"> </h1>
  <object
    classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swf
    lash.cab#version=7,0,0,0" width="1120" height="109">
    <param name="movie" value="../Media/Open_Possibilities.swf">
    <param name="quality" value="high">
    <embed
      src="../Media/Open_Possibilities.swf" quality="high"
    pluginspage="http://www.macromedia.com/go/getflashplayer"
    type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" width="1120"
    height="109"></embed>
  </object>
<?php
globalnav();
?>
</div>
<!--No navigation stuff for login page
<div id="navBar">
</div>
-->

<!--end headlines -->
<div id="content">
  <div class="feature">
    <h1>Create new IMAS user account</h1>
  </div>
<div class="story">

<?php echo "<FORM METHOD=\"POST\" action=\"create_user_success.php\"
name=\"userform\"><span style=\"color:#0000FF\"><h2>Please fill out the
fields below.</h2></span><br/>";

echo "<table><tr>";
echo "<td>First Name:</td><td><input type=\"text\" name=\"fname\"
onChange=\"emptyvalidation(fname, 'Please enter a first
name')\"/></td>";
echo "<td>Last Name:</td><td><input type=\"text\" name=\"lname\"
onChange=\"emptyvalidation(lname, 'Please enter a last
name')\"/></td></tr>";
echo "<tr><td>Login Name:</td><td><input type=\"text\" name=\"login\"
onChange=\"emptyvalidation(login, 'Please enter a login
name')\"/></td>";
echo "<td>Password:</td><td><input type=\"password\" name=\"pword\"
onChange=\"emptyvalidation(pword, 'Please enter a
password')\"/></td></tr>";
echo "<tr><td>Email:</td><td><input type=\"text\" name=\"email\"
onChange=\"emailvalidation(email, 'The E-mail is not valid')\"/></td>";
echo "<td>Email Password:</td><td><input type=\"text\"
name=\"email_pwd\" onChange=\"emptyvalidation(email_pwd, 'Please enter
your email password')\"/></td></tr>";
echo "<tr><td>Email Login Name:</td><td><input type=\"text\"
name=\"email_login\" onChange=\"emptyvalidation(email_login, 'Please
enter an email login name')\"/></td>";

```

```

echo      "<td>Email      IMAP      Server:</td><td><input      type=\"text\"
name=\"email_imap\"      onChange=\"emptyvalidation(email_imap,      'Please
enter your imap server')\"/></td></tr>";
echo      "<tr><td>IM      Screen-name:</td><td><input      type=\"text\"
name=\"im_sn\"      onChange=\"emptyvalidation(im_sn,      'Please enter your IM
screenname')\"/></td>";

require_once('../Connections/IMASInternalDatabase.php');
mysql_select_db($database_IMASInternalDatabase, $IMASInternalDatabase);
$query_Get_IM = "SELECT * FROM `im_provider` ";
$Get_IM      =      mysql_query($query_Get_IM,      $IMASInternalDatabase)      or
die(mysql_error());
echo "<td>IM Provider:</td><td><select name=\"im_provider\"/>";
while ($row = mysql_fetch_array($Get_IM)){
    $IM = $row["IM_provider"];
    echo "<option>$IM</option>";}
echo "</select></td></tr>";
echo "<tr><td>Cell # (10 digits only):</td><td><input type=\"text\"
name=\"cellphone\"      onChange=\"digitvalidation(cellphone,      10,      10,
'Please enter a 10 digit phone number', 'I')\"/></td>";

$query_Get_Cellprovider = "SELECT * FROM `cell_provider` ";
$Get_Cell = mysql_query($query_Get_Cellprovider, $IMASInternalDatabase)
or die(mysql_error());
echo "<td>Cellphone Provider</td><td><select name=\"cell_provider\"/>";
while ($row = mysql_fetch_array($Get_Cell)){
    $Cell = $row["cell_provider"];
    echo "<option>$Cell</option>";}
echo "</select></td></tr>";

//mysql_select_db($database_IMASInternalDatabase,
$IMASInternalDatabase);
$query_Get_Device = "SELECT `device_brand_number` FROM `device`";
$Get_Device = mysql_query($query_Get_Device, $IMASInternalDatabase) or
die(mysql_error());

echo "<tr><td>Cell Model:</td><td><select name=\"cellbrand\"/>";
while ($row = mysql_fetch_array($Get_Device)){
    $device = $row["device_brand_number"];
    echo "<option>$device</option>";}
echo "</select></td></tr>";
echo "<tr><td><INPUT TYPE = \"SUBMIT\" VALUE=\"Submit\"><INPUT TYPE =
\"RESET\" VALUE=\"reset\"></td></tr></table></form><br><br>";
?>
<script language="JavaScript" type="text/javascript">

    var frmvalidator  = new Validator("userform");
    frmvalidator.addValidation("fname","req","Please enter your First
Name");
    frmvalidator.addValidation("fname","alpha");

    frmvalidator.addValidation("lname","req","Please enter your Last
Name");
    frmvalidator.addValidation("lname","alpha");

    frmvalidator.addValidation("pword","req","Please enter a password");

```

```

frmvalidator.addValidation("pword","maxlen=32");

frmvalidator.addValidation("login","req","Please enter a login
name");
frmvalidator.addValidation("login","maxlen=25");

frmvalidator.addValidation("im_sn","req","Please enter a IM
screenname");
frmvalidator.addValidation("im_sn","maxlen=25");

frmvalidator.addValidation("email","maxlen=75");
frmvalidator.addValidation("email","req");
frmvalidator.addValidation("email","email");

frmvalidator.addValidation("email_pwd","req","Please enter an email
password");
frmvalidator.addValidation("email_pwd","maxlen=32");

frmvalidator.addValidation("email_login","req","Please enter an email
login name");
frmvalidator.addValidation("email_login","maxlen=32");

frmvalidator.addValidation("email_imap", "req", "Please enter you
email IMAP server");
frmvalidator.addValidation("email_imap","maxlen=75");

frmvalidator.addValidation("cellphone","maxlen=10");
frmvalidator.addValidation("cellphone","req");
frmvalidator.addValidation("cellphone","numeric");

</script>

<!-- Original Webcalendar Version Footer display - commented out
on February 28, 2006 -->
<a href="<?php echo $PROGRAM_URL ?>" id="programname"><?php echo
$PROGRAM_NAME?></a>

<?php // Print custom trailer (since we do not call print_trailer
function)
if ( ! empty ( $CUSTOM_TRAILER ) && $CUSTOM_TRAILER == 'Y' ) {
$res = dbi_query (
"SELECT cal_template_text FROM webcal_report_template " .
"WHERE cal_template_type = 'T' and cal_report_id = 0" );
if ( $res ) {
if ( $row = dbi_fetch_row ( $res ) ) {
echo $row[0];
}
dbi_free_result ( $res );
}
} ?>

</div>
</div>
<!--end content -->
<?php
siteInfo();
?>

```

```

<!--Site Info -->
<br>
</body>
</html>

```

B. CREATE_USER_SUCCESS.PHP

```

<?php
include "../Class/index_Nav.php";
include "includes/config.php";
include "includes/php-dbi.php";
include "includes/functions.php";
include "includes/$user_inc";
include "includes/connect.php";
include "../Connections/IMASInternalDatabase.php";
session_start();
$fname=$_POST['fname'];
$lname=$_POST['lname'];
$login=$_POST['login'];
$pword=md5($_POST['pword']);
$email=$_POST['email'];
$im_sn=$_POST['im_sn'];
$im_provider=$_POST['im_provider'];
$cellphone=$_POST['cellphone'];
$cell_provider=$_POST['cell_provider'];
$cellbrand=$_POST['cellbrand'];
$email_pwd=$_POST['email_pwd'];
$email_imap=$_POST['email_imap'];
$email_login=$_POST['email_login'];

//require_once('../Connections/IMASInternalDatabase.php');
mysql_select_db($database_IMASInternalDatabase, $IMASInternalDatabase);

$insert_into_webcal_user=      "INSERT      INTO      webcal_user
(cal_firstname,cal_lastname,cal_login,cal_passwd,   cal_email)  VALUES
(\"$fname\", \"$lname\", \"$login\", \"$pword\", \"$email\")";
$insert_into_webcal_user      =      mysql_query($insert_into_webcal_user,
$IMASInternalDatabase) or die(mysql_error());

$insert_into_email_account_tracking=      "INSERT      INTO
email_account_tracking(email_address,   cal_login,   email_password,
email_imap,   email_name)      VALUES
(\"$email\", \"$login\", \"$email_pwd\", \"$email_imap\",
 \"$email_login\")";
$insert_into_email_account_tracking      =
mysql_query($insert_into_email_account_tracking, $IMASInternalDatabase)
or die(mysql_error());

$insert_into_im_account_tracking=      "INSERT      INTO      im_account_tracking
(Screen_name,IM_provider,cal_login)      VALUES
(\"$im_sn\", \"$im_provider\", \"$login\")";
$insert_into_im_account_tracking      =
mysql_query($insert_into_im_account_tracking, $IMASInternalDatabase) or
die(mysql_error());

```

```

$insert_into_cellphone_account_tracking=          "INSERT          INTO
cellphone_account_tracking          (cellphone_number,device_brand_number,
cell_provider,cal_login)          VALUES
(\"$cellphone\", \"$cellbrand\", \"$cell_provider\", \"$login\");
$insert_into_cellphone_account_tracking          =
mysql_query($insert_into_cellphone_account_tracking,
$IMASInternalDatabase) or die(mysql_error());

```

```

//Original Webcalendar CSS Style Sheet - commented out on February 28,
2006

```

```

/* include "includes/styles.php"; */

```

```

// Print custom header (since we do not call print_header function)
if ( ! empty ( $CUSTOM_SCRIPT ) && $CUSTOM_SCRIPT == 'Y' ) {
    $res = dbi_query (
        "SELECT cal_template_text FROM webcal_report_template " .
        "WHERE cal_template_type = 'S' and cal_report_id = 0" );
    if ( $res ) {
        if ( $row = dbi_fetch_row ( $res ) ) {
            echo $row[0];
        }
        dbi_free_result ( $res );
    }
}
?>

```

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<!-- DW6 -->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<title>IMAS Portal</title>
<link rel="stylesheet" href="../../CSS_Styles/Portal_Public.css"
type="text/css">
</head>

```

```

<script type="text/javascript">
// error check login/password

```

```

function myOnLoad() {
    <?php if ( ! empty ( $plugins_enabled ) && ( $plugins_enabled ) ){ ?>
        if (self != top) {
            window.open("login.php","_top","");
            return;
        }
    <?php } ?>
    document.login_form.login.focus();
    <?php
        if ( ! empty ( $login ) ) echo
"document.login_form.login.select();"
        if ( ! empty ( $error ) ) {
            echo " alert ( \"$error\" );\n";
        }
    ?>
}

```

```

}
</script>

<body onload="myOnLoad();">
<div id="masthead">
  <h1 id="siteName"> </h1>
  <object
    classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swf
    lash.cab#version=7,0,0,0" width="1120" height="109">
    <param name="movie" value="../../Media/Open_Possibilities.swf">
    <param name="quality" value="high">
    <embed
      src="../../Media/Open_Possibilities.swf" quality="high"
      pluginpage="http://www.macromedia.com/go/getflashplayer"
      type="application/x-shockwave-flash"
      pluginpage="http://www.macromedia.com/go/getflashplayer" width="1120"
      height="109"></embed>
    </object>

  <?php
    globalnav();
  ?>

</div>

<!--end headlines -->
<div id="content">
  <div class="feature">
    <h1>Congratulations on successfully creating an IMAS account!</h1>
    <p style="color:#0000FF">Use the above menu to navigate
    IMAS</p><br/>

  </div>

  <!-- Original Webcalendar Version Footer display - commented out
  on February 28, 2006 -->
  <a href="<?php echo $PROGRAM_URL ?>" id="programname"><?php echo
  $PROGRAM_NAME?></a>

  <?php // Print custom trailer (since we do not call print_trailer
  function)
  if ( ! empty ( $CUSTOM_TRAILER ) && $CUSTOM_TRAILER == 'Y' ) {
    $res = dbi_query (
      "SELECT cal_template_text FROM webcal_report_template " .
      "WHERE cal_template_type = 'T' and cal_report_id = 0" );
    if ( $res ) {
      if ( $row = dbi_fetch_row ( $res ) ) {
        echo $row[0];
      }
      dbi_free_result ( $res );
    }
  } ?>

</p>
</div>

```

```

</div>
<!--end content -->
<?php
siteInfo();
?>
<!--Site Info -->
<br>
</body>
</html>

```

C. **EDIT_ENTRY.PHP**

```

<?php
/*
 * $Id: edit_entry.php,v 1.91.2.1 2005/08/08 23:30:57 cknudsen Exp $
 *
 * Description:
 * Presents page to edit/add an event
 *
 * Notes:
 * If htmlarea is installed, users can use WYSIWYG editing.
 * SysAdmin must enable HTML for event full descriptions.
 * The htmlarea files should be installed so that the htmlarea.php
 * file is in ../includes/htmlarea/htmlarea.php
 * The htmlarea code can be downloaded from:
 * http://www.htmlarea.com
 * TODO
 * This file will not pass XHTML validation with HTMLArea enabled
 */
include_once 'includes/init.php';
include_once 'includes/site_extras.php';

load_user_categories ();

// Default for using tabs is enabled
if ( empty ( $EVENT_EDIT_TABS ) )
    $EVENT_EDIT_TABS = 'Y'; // default
$useTabs = ( $EVENT_EDIT_TABS == 'Y' );

// make sure this is not a read-only calendar
$can_edit = false;

// Public access can only add events, not edit.
if ( $login == "__public__" && $id > 0 ) {
    $id = 0;
}

$external_users = "";
$participants = array ();

if ( $readonly == 'Y' ) {
    $can_edit = false;
} else if ( ! empty ( $id ) && $id > 0 ) {
    // first see who has access to edit this entry
    if ( $is_admin ) {

```



```

$scan_edit = true;
} else {
$scan_edit = false;
if ( $readonly == "N" || $is_admin ) {
    $sql = "SELECT webcal_entry.cal_id FROM webcal_entry, " .
        "webcal_entry_user WHERE webcal_entry.cal_id = " .
        "webcal_entry_user.cal_id AND webcal_entry.cal_id = $id " .
        "AND (webcal_entry.cal_create_by = '$login' " .
        "OR webcal_entry_user.cal_login = '$login')";
    $res = dbi_query ( $sql );
    if ( $res ) {
        $row = dbi_fetch_row ( $res );
        if ( $row && $row[0] > 0 )
            $scan_edit = true;
        dbi_free_result ( $res );
    }
}
}
$sql = "SELECT cal_create_by, cal_date, cal_time, cal_mod_date, " .
    "cal_mod_time, cal_duration, cal_priority, cal_type, cal_access, " .
    "cal_name, cal_description, cal_group_id FROM webcal_entry WHERE
cal_id = " . $id;
$res = dbi_query ( $sql );
if ( $res ) {
    $row = dbi_fetch_row ( $res );
    if ( ! empty ( $override ) && ! empty ( $date ) ) {
        // Leave $cal_date to what was set in URL with date=YYYYMMDD
        $cal_date = $date;
    } else {
        $cal_date = $row[1];
    }
    $create_by = $row[0];
    if ( ( $user == $create_by ) && ( $is_assistant || $is_nonuser_admin
)) $scan_edit = true;

    $year = (int) ( $cal_date / 10000 );
    $month = ( $cal_date / 100 ) % 100;
    $day = $cal_date % 100;
    $time = $row[2];
    // test for AllDay event, if so, don't adjust time
    if ( $time > 0 || ( $time == 0 && $row[5] != 1440 ) ) { /* -1 =
no time specified */
        $time += ( ! empty ( $TZ_OFFSET ) ? $TZ_OFFSET : 0 ) * 10000;
        if ( $time > 240000 ) {
            $time -= 240000;
            $gmt = mktime ( 3, 0, 0, $month, $day, $year );
            $gmt += $ONE_DAY;
            $month = date ( "m", $gmt );
            $day = date ( "d", $gmt );
            $year = date ( "Y", $gmt );
        } else if ( $time < 0 ) {
            $time += 240000;
            $gmt = mktime ( 3, 0, 0, $month, $day, $year );
            $gmt -= $ONE_DAY;
            $month = date ( "m", $gmt );
            $day = date ( "d", $gmt );

```

```

        $year = date ( "Y", $gmt );
    }
    // Set alerted date
    $scal_date = sprintf("%04d%02d%02d",$year,$month,$day);
}
if ( $time >= 0 ) {
    $hour = floor($time / 10000);
    $minute = ( $time / 100 ) % 100;
    $duration = $row[5];
} else {
    $duration = "";
    $hour = -1;
}
$priority = $row[6];
$type = $row[7];
$access = $row[8];
$name = $row[9];
$description = $row[10];
$parent = $row[11];
// check for repeating event info...
// but not if we are overriding a single entry of an already
repeating
// event... confusing, eh?
if ( ! empty ( $override ) ) {
    $rpt_type = "none";
    $rpt_end = 0;
    $rpt_end_date = $scal_date;
    $rpt_freq = 1;
    $rpt_days = "nnnnnnn";
    $rpt_sun = $rpt_mon = $rpt_tue = $rpt_wed =
        $rpt_thu = $rpt_fri = $rpt_sat = false;
} else {
    $res = dbi_query ( "SELECT cal_id, cal_type, cal_end, " .
        "cal_frequency, cal_days FROM webcal_entry_repeats " .
        "WHERE cal_id = $id" );
    if ( $res ) {
        if ( $row = dbi_fetch_row ( $res ) ) {
            $rpt_type = $row[1];
            if ( $row[2] > 0 )
                $rpt_end = date_to_epoch ( $row[2] );
            else
                $rpt_end = 0;
            $rpt_end_date = $row[2];
            $rpt_freq = $row[3];
            $rpt_days = $row[4];
            $rpt_sun = ( substr ( $rpt_days, 0, 1 ) == 'y' );
            $rpt_mon = ( substr ( $rpt_days, 1, 1 ) == 'y' );
            $rpt_tue = ( substr ( $rpt_days, 2, 1 ) == 'y' );
            $rpt_wed = ( substr ( $rpt_days, 3, 1 ) == 'y' );
            $rpt_thu = ( substr ( $rpt_days, 4, 1 ) == 'y' );
            $rpt_fri = ( substr ( $rpt_days, 5, 1 ) == 'y' );
            $rpt_sat = ( substr ( $rpt_days, 6, 1 ) == 'y' );
        }
    }
}
}
}

```

```

    $sql = "SELECT cal_login, cal_category FROM webcal_entry_user WHERE
cal_id = $id";
    $res = dbi_query ( $sql );
    if ( $res ) {
        while ( $row = dbi_fetch_row ( $res ) ) {
            $participants[$row[0]] = 1;
            if ($login == $row[0]) $cat_id = $row[1];
            if ( ( $is_assistant || $is_admin ) && $user == $row[0]) $cat_id
= $row[1];
        }
    }
    if ( ! empty ( $allow_external_users ) && $allow_external_users ==
"Y" ) {
        $external_users = event_get_external_users ( $id );
    }
} else {
    // New event.
    $id = 0; // to avoid warnings below about use of undefined var
    // Anything other then testing for strlen breaks either hour=0 or no
hour in URL
    if ( strlen ( $hour ) ) {
        $time = $hour * 100;
    } else {
        $time = -1;
        $hour = -1;
    }
    if ( ! empty ( $defusers ) ) {
        $tmp_ar = explode ( ",", $defusers );
        for ( $i = 0; $i < count ( $tmp_ar ); $i++ ) {
            $participants[$tmp_ar[$i]] = 1;
        }
    }
    if ( $readonly == "N" ) {
        // If public, then make sure we can add events
        if ( $login == '__public__' ) {
            if ( $public_access_can_add )
                $can_edit = true;
        } else {
            // not public user
            $can_edit = true;
        }
    }
}
}
if ( ! empty ( $year ) && $year )
    $thisyear = $year;
if ( ! empty ( $month ) && $month )
    $thismonth = $month;
if ( ! empty ( $day ) && $day )
    $thisday = $day;
if ( empty ( $rpt_type ) || ! $rpt_type )
    $rpt_type = "none";

// avoid error for using undefined vars
if ( ! isset ( $hour ) )
    $hour = -1;
if ( empty ( $duration ) )
    $duration = 0;

```

```

if ( $duration == ( 24 * 60 ) ) {
    $hour = $minute = $duration = "";
    $allday = "Y";
} else
    $allday = "N";
if ( empty ( $name ) )
    $name = "";
if ( empty ( $description ) )
    $description = "";
if ( empty ( $priority ) )
    $priority = 0;
if ( empty ( $access ) )
    $access = "";
if ( empty ( $rpt_freq ) )
    $rpt_freq = 0;
if ( empty ( $rpt_end_date ) )
    $rpt_end_date = 0;

if ( ( empty ( $year ) || ! $year ) &&
    ( empty ( $month ) || ! $month ) &&
    ( ! empty ( $date ) && strlen ( $date ) ) ) {
    $thisyear = $year = substr ( $date, 0, 4 );
    $thismonth = $month = substr ( $date, 4, 2 );
    $thisday = $day = substr ( $date, 6, 2 );
    $cal_date = $date;
} else {
    if ( empty ( $cal_date ) )
        $cal_date = date ( "Ymd" );
}
if ( empty ( $thisyear ) )
    $thisdate = date ( "Ymd" );
else {
    $thisdate = sprintf ( "%04d%02d%02d",
        empty ( $thisyear ) ? date ( "Y" ) : $thisyear,
        empty ( $thismonth ) ? date ( "m" ) : $thismonth,
        empty ( $thisday ) ? date ( "d" ) : $thisday );
}
if ( empty ( $cal_date ) || ! $cal_date )
    $cal_date = $thisdate;

if ( $allow_html_description == "Y" ){
    // Allow HTML in description
    // If they have installed the htmlarea widget, make use of it
    $textareasize = 'rows="15" cols="50"';
    if ( file_exists ( "includes/htmlarea/htmlarea.php" ) ) {
        $BodyX
        'onload="initEditor();timetype_handler();rpttype_handler()"' ;
        $INC = array ( 'htmlarea/htmlarea.php', 'js/edit_entry.php',
            'js/visible.php', 'htmlarea/core.php' );
    } else {
        // No htmlarea files found...
        $BodyX = 'onload="timetype_handler();rpttype_handler()"' ;
        $INC = array ( 'js/edit_entry.php', 'js/visible.php' );
    }
} else {
    $textareasize = 'rows="5" cols="40"';
    $BodyX = 'onload="timetype_handler();rpttype_handler()"' ;

```

```

    $INC = array('js/edit_entry.php','js/visible.php');
}

print_header ( $INC, '', $BodyX );
?>

<h2><?php if ( $id ) echo translate("Edit Entry"); else echo
translate("Add Entry"); ?>
    &nbsp;  "
class="help" onclick="window.open ( 'help_edit_entry.php<?php if (
empty ( $id ) ) echo "?add=1"; ?>', 'cal_help',
'dependent,menubar,scrollbars,height=400,width=400,innerHeight=420,outer
Width=420');" /></h2>
<br>
<br>
<br>
<?php
    if ( $can_edit ) {
?>
<form          action="edit_entry_handler.php"          method="post"
name="editentryform">

<?php
if ( ! empty ( $id ) && ( empty ( $copy ) || $copy != '1' ) ) echo
"<input type=\"hidden\" name=\"id\" value=\"\$id\" />\n";
// we need an additional hidden input field
echo "<input type=\"hidden\" name=\"entry_changed\" value=\"\" />\n";

// are we overriding an entry from a repeating event...
if ( ! empty ( $override ) ) {
    echo "<input type=\"hidden\" name=\"override\" value=\"1\" />\n";
    echo "    <input          type=\"hidden\"          name=\"override_date\"
value=\"\$cal_date\" />\n";
}
// if assistant, need to remember boss = user
if ( $is_assistant || $is_nonuser_admin || ! empty ( $user ) )
    echo "<input type=\"hidden\" name=\"user\" value=\"\$user\" />\n";

// if has cal_group_id was set, need to send parent = $parent
if ( ! empty ( $parent ) )
    echo "    <input type=\"hidden\" name=\"parent\" value=\"\$parent\"
/>\n";

?>

<!-- TABS -->
<?php if ( $useTabs ) { ?>
<div id="tabs">
    <span class="tabfor" id="tab_details"><a href="#tabdetails"
onclick="return showTab('details')"><?php etranslate("Details")
?></a></span>
    <?php if ( $disable_participants_field != "Y" ) { ?>
        <span class="tabbak" id="tab_participants"><a
href="#tabparticipants" onclick="return showTab('participants')"><?php
etranslate("Participants") ?></a></span>
    <?php } ?>

```

```

    <?php if ( $disable_repeating_field != "Y" ) { ?>
        <span class="tabbak" id="tab_pete"><a href="#tabpete"
onclick="return showTab('pete')"><?php etranslate("Repeat")
?></a></span>
    <?php } ?>
</div>
<?php } ?>

<!-- TABS BODY -->
<?php if ( $useTabs ) { ?>
<div id="tabscontent">
    <!-- DETAILS -->
    <a name="tabdetails"></a>
    <div id="tabscontent_details">
<?php } ?>
    <table style="border-width:0px;">
        <tr><td style="width:14%;" class="tooltip" title="<?php
etooltip("brief-description-help")?>">
            <label for="entry_brief"><?php etranslate("Brief
Description")?></label></td><td>
                <input type="text" name="name" id="entry_brief" size="25"
value="<?php
                echo htmlspecialchars ( $name );
                ?>" /></td><td style="width:35%;">
            </td></tr>
        <tr><td style="vertical-align:top;" class="tooltip" title="<?php
etooltip("full-description-help")?>">
            <label for="entry_full"><?php etranslate("Full
Description")?></label></td><td>
                <textarea name="description" id="entry_full" <?php
                echo $textareasize;
                ?><?php
                echo htmlspecialchars ( $description );
                ?></textarea></td><td style="vertical-align:top;">

<?php if ( ( ! empty ( $categories ) ) || ( $disable_access_field != "Y"
) ||
    ( $disable_priority_field != "Y" ) ){ // new table for extra
fields ?>
    <table>
<?php } ?>
<?php if ( $disable_access_field != "Y" ) { ?>
    <tr><td class="tooltip" title="<?php etooltip("access-help")?>">
        <label for="entry_access">
        <?php etranslate("Context")?>
        </label></td><td>
            <select name="access" id="entry_access">
                <option value="G"><?php if ( $access== "G" || ! strlen ( $access
) ) echo "selected=\"selected\"";?><?php
etranlate("General")?></option>
                <option value="M"><?php if ( $access== "M" ) echo "
selected=\"selected\"";?><?php etranslate("Meeting")?></option>
                <option value="D"><?php if ( $access== "D" ) echo "
selected=\"selected\"";?><?php etranslate("Do not disturb")?></option>
            </select>
        </td></tr>
<?php } ?>

```

```

<?php if ( $disable_priority_field != "Y" ) { ?>
    <tr><td class="tooltip" title="<?php etooltip("priority-help")?>">
        <label
            for="entry_alert"><?php
                etranslate("Alert
Type")?>: &nbsp;</label></td><td>
            <select name="priority" id="entry_priority">
                <option value="1"><?php if ( $priority == 1 ) echo "
selected=\"selected\"";?><?php etranslate("Immediate")?></option>
                <option value="2"><?php if ( $priority == 2 ) echo "
selected=\"selected\"";?><?php etranslate("Scheduled")?></option>
                <option value="3"><?php if ( $priority == 3 ) echo "
selected=\"selected\"";?><?php etranslate("Subscription")?></option>
                <option value="4"><?php if ( $priority == 4 ) echo "
selected=\"selected\"";?><?php etranslate("Location")?></option>
            </select>
        </td></tr>
<?php } ?>
<?php if ( ! empty ( $categories ) ) { ?>
    <tr><td class="tooltip" title="<?php etooltip("category-help")?>">
        <label
            for="entry_categories"><?php
                etranslate("Category")?>: &nbsp;</label></td><td>
            <select name="cat_id" id="entry_categories">
                <option value=""><?php etranslate("None")?></option>
                <?php
                    foreach( $categories as $K => $V ){
                        echo "
                        <option value=\"\$K\"";
                        if ( $cat_id == $K ) echo " selected=\"selected\"";
                        echo ">\$V</option>\n";
                    }
                ?>
            </select>
        </td></tr>
<?php } //end if (! empty ($categories)) ?>
<?php if ( ( ! empty ( $categories ) ) || ( $disable_access_field != "Y"
) ||
    ( $disable_priority_field != "Y" ) ){ // end the table ?>
    </table>

<?php } ?>
</td></tr>
<tr><td class="tooltip" title="<?php etooltip("date-help")?>">
    <?php etranslate("Date")?>:</td><td colspan="2">
        <?php
            print_date_selection ( "", $cal_date );
        ?>
    </td></tr>
<tr><td>&nbsp;</td><td colspan="2">
    <select name="timetype" onchange="timetype_handler()">
        <option value="U"><?php if ( $allday != "Y" && $hour == -1 ) echo "
selected=\"selected\""?><?php etranslate("Untimed event"); ?></option>
        <option value="T"><?php if ( $allday != "Y" && $hour >= 0 ) echo "
selected=\"selected\""?><?php etranslate("Timed event"); ?></option>
        <option value="A"><?php if ( $allday == "Y" ) echo "
selected=\"selected\""?><?php etranslate("All day event"); ?></option>
    </select>
</td></tr>
<tr
    id="timeentrystart"><td
        class="tooltip"
        title="<?php
            etooltip("time-help")?>">

```

```

        <?php echo translate("Time") . ":"; ?></td><td colspan="2">
<?php
$h12 = $hour;
$samsel = " checked=\"checked\""; $pmsel = "";
if ( $TIME_FORMAT == "12" ) {
    if ( $h12 < 12 ) {
        $samsel = " checked=\"checked\""; $pmsel = "";
    } else {
        $samsel = ""; $pmsel = " checked=\"checked\"";
    }
    $h12 %= 12;
    if ( $h12 == 0 ) $h12 = 12;
}
if ( $time < 0 )
    $h12 = "";
?>
    <input type="text" name="hour" size="2" value="<?php
        if ( $time >= 0 && $allday != 'Y' ) echo $h12;
    ?>" maxlength="2" />:<input type="text" name="minute" size="2"
value="<?php
    if ( $time >= 0 && $allday != "Y" ) printf ( "%02d", $minute );
    ?>" maxlength="2" />
<?php
if ( $TIME_FORMAT == "12" ) {
    echo "<label><input type=\"radio\" name=\"ampm\" value=\"am\" $samsel
/>&nbsp;";
    translate("am") . "</label>\n";
    echo "<label><input type=\"radio\" name=\"ampm\" value=\"pm\" $pmsel
/>&nbsp;";
    translate("pm") . "</label>\n";
}
?>

<?php
$dur_h = (int)( $duration / 60 );
$dur_m = $duration - ( $dur_h * 60 );

if ( $GLOBALS['TIMED_EVT_LEN'] != 'E' ) { ?>
    </td></tr>
    <tr id="timeentryduration"><td>
    <span class="tooltip" title="<?php
        etooltip("duration-help")
    ?>"><?php
        etranslate("Duration")
    ?>:&nbsp;</span></td><td colspan="2">
    <input type="text" name="duration_h" id="duration_h" size="2"
maxlength="2" value="<?php
        if ( $allday != "Y" ) printf ( "%d", $dur_h );
    ?>" />:<input type="text" name="duration_m" id="duration_m" size="2"
maxlength="2" value="<?php
        if ( $allday != "Y" )
            printf ( "%02d", $dur_m );
    ?>" />&nbsp;<label for="duration_h"><?php
        echo translate("hours")
    ?></label>:<label for="duration_m"><?php
        echo translate("minutes")
    ?></label>

```



```

</span>
</td></tr>
<?php } ?>
</table>
<table>
<?php
// site-specific extra fields (see site_extras.php)
// load any site-specific fields and display them
if ( $id > 0 )
    $extras = get_site_extra_fields ( $id );
for ( $i = 0; $i < count ( $site_extras ); $i++ ) {
    $extra_name = $site_extras[$i][0];
    $extra_descr = $site_extras[$i][1];
    $extra_type = $site_extras[$i][2];
    $extra_arg1 = $site_extras[$i][3];
    $extra_arg2 = $site_extras[$i][4];
    //echo "<tr><td>Extra " . $extra_name . " - " . $site_extras[$i][2] .
    // " - " . $extras[$extra_name]['cal_name'] .
    // "arg1: $extra_arg1, arg2: $extra_arg2 </td></tr>\n";
    if ( $extra_type == $EXTRA_MULTILINETEXT )
        echo "<tr><td style=\"vertical-align:top; font-weight:bold;\"><br
/>\n";
    else
        echo "<tr><td style=\"font-weight:bold;\">";
        echo translate ( $extra_descr ) . " :</td><td>\n";
        if ( $extra_type == $EXTRA_URL ) {
            echo "<input type=\"text\" size=\"50\" name=\"\" . $extra_name .
            \"\" value=\"\" . ( empty ( $extras[$extra_name]['cal_data'] ) ) ?
            \"\" : htmlspecialchars ( $extras[$extra_name]['cal_data'] ) ) .
            \"\" />";
        } else if ( $extra_type == $EXTRA_EMAIL ) {
            echo "<input type=\"text\" size=\"30\" name=\"\" . $extra_name . \"\"
            value=\"\" . ( empty ( $extras[$extra_name]['cal_data'] ) ) ?
            \"\" : htmlspecialchars ( $extras[$extra_name]['cal_data'] ) ) .
            \"\" />";
        } else if ( $extra_type == $EXTRA_DATE ) {
            if ( ! empty ( $extras[$extra_name]['cal_date'] ) )
                print_date_selection ( $extra_name,
                $extras[$extra_name]['cal_date'] );
            else
                print_date_selection ( $extra_name, $cal_date );
        } else if ( $extra_type == $EXTRA_TEXT ) {
            $size = ( $extra_arg1 > 0 ? $extra_arg1 : 50 );
            echo "<input type=\"text\" size=\"\" . $size . \"\" name=\"\" .
            $extra_name .
            \"\" value=\"\" . ( empty ( $extras[$extra_name]['cal_data'] ) ) ?
            \"\" : htmlspecialchars ( $extras[$extra_name]['cal_data'] ) ) .
            \"\" />";
        } else if ( $extra_type == $EXTRA_MULTILINETEXT ) {
            $cols = ( $extra_arg1 > 0 ? $extra_arg1 : 50 );
            $rows = ( $extra_arg2 > 0 ? $extra_arg2 : 5 );
            echo "<textarea rows=\"\" . $rows . \"\" cols=\"\" . $cols . \"\"
            name=\"\" . $extra_name . \"\">\" . ( empty (
            $extras[$extra_name]['cal_data'] ) ) ?
            \"\" : htmlspecialchars ( $extras[$extra_name]['cal_data'] ) ) .
            \"</textarea>";
        } else if ( $extra_type == $EXTRA_USER ) {

```

```

// show list of calendar users...
echo "<select name=\"\" . $extra_name . \">\n";
echo "<option value=\"\">None</option>\n";
$userlist = get_my_users ();
for ( $j = 0; $j < count ( $userlist ); $j++ ) {
    echo "<option value=\"\" . $userlist[$j]['cal_login'] . \"\"";
    if ( ! empty ( $extras[$extra_name]['cal_data'] ) &&
        $userlist[$j]['cal_login']
$extras[$extra_name]['cal_data'] )
        echo " selected=\"selected\"";
    echo ">\" . $userlist[$j]['cal_fullname'] . "</option>\n";
}
echo "</select>\n";
} else if ( $extra_type == $EXTRA_REMINDER ) {
    $rem_status = 0; // don't send
    echo "<label><input type=\"radio\" name=\"\" . $extra_name . \"\"
value=\"1\"\"";
    if ( empty ( $id ) ) {
        // adding event... check default
        if ( ( $extra_arg2 & $EXTRA_REMINDER_DEFAULT_YES ) > 0 )
            $rem_status = 1;
    } else {
        // editing event... check status
        if ( ! empty ( $extras[$extra_name]['cal_remind'] ) )
            $rem_status = 1;
    }
    if ( $rem_status )
        echo " checked=\"checked\"";
    echo " />";
    etranslate ( "Yes" );
    echo "</label>&nbsp;<label><input type=\"radio\" name=\"\" .
$extra_name . \"\" value=\"0\"\"";
    if ( ! $rem_status )
        echo " checked=\"checked\"";
    echo " />";
    etranslate ( "No" );
    echo "</label>&nbsp;&nbsp;&nbsp;";
    if ( ( $extra_arg2 & $EXTRA_REMINDER_WITH_DATE ) > 0 ) {
        if ( ! empty ( $extras[$extra_name]['cal_date'] ) &&
            $extras[$extra_name]['cal_date'] > 0 )
            print_date_selection ( $extra_name,
$extras[$extra_name]['cal_date'] );
        else
            print_date_selection ( $extra_name, $cal_date );
    } else if ( ( $extra_arg2 & $EXTRA_REMINDER_WITH_OFFSET ) > 0 ) {
        if ( ! empty ( $extras[$extra_name]['cal_data'] ) )
            $minutes = $extras[$extra_name]['cal_data'];
        else
            $minutes = $extra_arg1;
        // will be specified in total minutes
        $d = (int) ( $minutes / ( 24 * 60 ) );
        $minutes -= ( $d * 24 * 60 );
        $h = (int) ( $minutes / 60 );
        $minutes -= ( $h * 60 );
        echo "<label><input type=\"text\" size=\"2\" name=\"\" .
$extra_name .

```

```

        "_days\" value=\"\$d\" /> " . translate("days") .
"</label>&nbsp;\n";
        echo "<label><input type=\"text\" size=\"2\" name=\"" .
$extra_name .
        "_hours\" value=\"\$h\" /> " . translate("hours") .
"</label>&nbsp;\n";
        echo "<label><input type=\"text\" size=\"2\" name=\"" .
$extra_name .
        "_minutes\" value=\"\$minutes\" /> " . translate("minutes") .
"&nbsp;\n";
        translate("before event") . "</label>";
    }
} else if ( $extra_type == $EXTRA_SELECTLIST ) {
    // show custom select list.
    echo "<select name=\"" . $extra_name . "\">\n";
    if ( is_array ( $extra_arg1 ) ) {
        for ( $j = 0; $j < count ( $extra_arg1 ); $j++ ) {
            echo "<option";
            if ( ! empty ( $extras[$extra_name]['cal_data'] ) &&
                $extra_arg1[$j] == $extras[$extra_name]['cal_data'] )
                echo " selected=\"selected\"";
            echo ">" . $extra_arg1[$j] . "</option>\n";
        }
    }
    echo "</select>\n";
}
echo "</td></tr>\n";
}
// end site-specific extra fields
?>
</table>
<?php if ( $useTabs ) { ?> test
</div>
<?php } /* $useTabs */ ?>

<!-- PARTICIPANTS -->
<?php if ( $useTabs ) { ?>
<a name="tabparticipants"></a>
<div id="tabscontent_participants">
<?php } /* $useTabs */ ?>
<table>
<?php
// Only ask for participants if we are multi-user.
$show_participants = ( $disable_participants_field != "Y" );
if ( $is_admin )
    $show_participants = true;
if ( $login == "__public__" && $public_access_others != "Y" )
    $show_participants = false;

if ( $single_user == "N" && $show_participants ) {
    $userlist = get_my_users ();
    if ( $nonuser_enabled == "Y" ) {
        $nonusers = get_nonuser_cals ();
        $userlist = ( $nonuser_at_top == "Y" ) ? array_merge($nonusers,
$userlist) : array_merge($userlist, $nonusers);
    }
    $num_users = 0;
    $size = 0;

```

```

$users = "";
for ( $i = 0; $i < count ( $userlist ); $i++ ) {
    $l = $userlist[$i]['cal_login'];
    $size++;
    $users .= "<option value=\"\" . $l . \"\"";
    if ( $id > 0 ) {
        if ( ! empty ( $participants[$l] ) )
            $users .= " selected=\"selected\"";
    } else {
        if ( ! empty ( $defusers ) ) {
            // default selection of participants was in the URL
            if ( ! empty ( $participants[$l] ) )
                $users .= " selected=\"selected\"";
        } else {
            if ( ($l == $login && ! $is_assistant && ! $is_nonuser_admin)
|| (! empty ( $user ) && $l == $user) )
                $users .= " selected=\"selected\"";
        }
        if ( $l == '__public__' &&
! empty ( $public_access_default_selected ) &&
$public_access_default_selected == 'Y' )
            $users .= " selected=\"selected\"";
    }
    $users .= ">" . $userlist[$i]['cal_fullname'] . "</option>\n";
}

if ( $size > 50 )
    $size = 15;
else if ( $size > 5 )
    $size = 5;
print "<tr title=\"\" .
tooltip("participants-help") . "\"><td
class=\"tooltipselect\">\n<label for=\"entry_part\">" .
translate("Participants") . " :</label></td><td>\n";
print "<select name=\"participants[]\" id=\"entry_part\"
size=\"$size\" multiple=\"multiple\">$users\n";
print "</select>\n";
if ( $groups_enabled == "Y" ) {
    echo "<input type=\"button\" onclick=\"selectUsers()\" value=\"\" .
        translate("Select") . "...\" />\n";
}
echo "<input type=\"button\" onclick=\"showSchedule()\" value=\"\" .
    translate("Availability") . "...\" />\n";
print "</td></tr>\n";

// external users
if ( ! empty ( $allow_external_users ) && $allow_external_users ==
"Y" ) {
    print "<tr title=\"\" .
        tooltip("external-participants-help") . "\"><td style=\"vertical-
align:top;\" class=\"tooltip\">\n<label for=\"entry_extpart\">" .
        translate("External Participants") . " :</label></td><td>\n";
    print "<textarea name=\"externalparticipants\" id=\"entry_extpart\"
rows=\"5\" cols=\"40\">";
    print $external_users . "</textarea>\n</td></tr>\n";
}
}

```

```

?>
</table>
<?php if ( $useTabs ) { ?>
</div>
<?php } /* $useTabs */ ?>

<!-- REPEATING INFO -->
<?php if ( $disable_repeating_field != "Y" ) { ?>
<?php if ( $useTabs ) { ?>
<a name="tabpete"></a>
<div id="tabscontent_pete">
<?php } /* $useTabs */ ?>
<table>
<tr style="vertical-align:top;"><td class="tooltip" title="<?php
etooltip("repeat-type-help")?>">
<label for="rpttype"><?php etranslate("Repeat
Type")?></label></td><td>
<select name="rpt_type" id="rpttype" onchange="rpttype_handler()">
<?php
echo " <option value=\"none\"\" .
( strcmp ( $rpt_type, 'none' ) == 0 ? " selected=\"selected\"\" : "" )
. ">" .
translate("None") .
"</option>\n";
echo " <option value=\"daily\"\" .
( strcmp ( $rpt_type, 'daily' ) == 0 ? " selected=\"selected\"\" : ""
) . ">" .
translate("Daily") .
"</option>\n";
echo " <option value=\"weekly\"\" .
( strcmp ( $rpt_type, 'weekly' ) == 0 ? " selected=\"selected\"\" : ""
) . ">" .
translate("Weekly") .
"</option>\n";
echo " <option value=\"monthlyByDay\"\" .
( strcmp ( $rpt_type, 'monthlyByDay' ) == 0 ? "
selected=\"selected\"\" : "" ) . ">" .
translate("Monthly") . " (" . translate("by day") . ")" . "
</option>\n";
echo " <option value=\"monthlyByDayR\"\" .
( strcmp ( $rpt_type, 'monthlyByDayR' ) == 0 ? "
selected=\"selected\"\" : "" ) . ">" .
translate("Monthly") . " (" . translate("by day (from end)") . ")" . "
"</option>\n";
echo " <option value=\"monthlyByDate\"\" .
( strcmp ( $rpt_type, 'monthlyByDate' ) == 0 ? "
selected=\"selected\"\" : "" ) . ">" .
translate("Monthly") . " (" . translate("by date") . ")" . "
"</option>\n";
echo " <option value=\"yearly\"\" .
( strcmp ( $rpt_type, 'yearly' ) == 0 ? " selected=\"selected\"\" : ""
) . ">" .
translate("Yearly") .
"</option>\n";
?>
</select>
</td></tr>

```

```
|  |  |  | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| id="rptenddate" style="visibility:hidden;"><td class="tooltip" title="<?php etooltip("repeat-end-date-help")??>"><?php etranslate("Repeat End Date")?:</td><td><label><input type="checkbox" name="rpt_end_use" value="y" <?php echo ( ! empty ( $rpt_end ) ? " checked=\"checked\" " : " " ); ?> />&nbsp;<?php etranslate("Use end date")??</label>&nbsp;&nbsp;&nbsp;<span class="end_day_selection"><?php print_date_selection ( "rpt_", $rpt_end_date ? $rpt_end_date : $scal_date ) ?></span></td></tr> |  |  | | --- | --- | | id="rptfreq" style="visibility:hidden;" title="<?php etooltip("repeat-frequency-help")??>"><td class="tooltip"><label for="entry_freq"><?php etranslate("Frequency")?:</label></td><td><input type="text" name="rpt_freq" id="entry_freq" size="4" maxlength="4" value="<?php echo $rpt_freq; ?>" /></td></tr> |  | | --- | | id="rptday" style="visibility:hidden;" title="<?php etooltip("repeat-day-help")??>"><td class="tooltip"><?php etranslate("Repeat Day")?:&nbsp; </td><td><?php if( $WEEK_START != 1) echo "<label><input type=\"checkbox\" name=\"rpt_sun\" value=\"y\\\" . (!empty($rpt_sun)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Sunday") . "</label>\n"; echo "<label><input type=\"checkbox\" name=\"rpt_mon\" value=\"y\\\" . (!empty($rpt_mon)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Monday") . "</label>\n"; echo "<label><input type=\"checkbox\" name=\"rpt_tue\" value=\"y\\\" . (!empty($rpt_tue)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Tuesday") . "</label>\n"; echo "<label><input type=\"checkbox\" name=\"rpt_wed\" value=\"y\\\" . (!empty($rpt_wed)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Wednesday") . "</label>\n"; echo "<label><input type=\"checkbox\" name=\"rpt_thu\" value=\"y\\\" . (!empty($rpt_thu)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Thursday") . "</label>\n"; echo "<label><input type=\"checkbox\" name=\"rpt_fri\" value=\"y\\\" . (!empty($rpt_fri)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Friday") . "</label>\n"; echo "<label><input type=\"checkbox\" name=\"rpt_sat\" value=\"y\\\" . (!empty($rpt_sat)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Saturday") . "</label>\n"; if( $WEEK_START == 1) echo "<label><input type=\"checkbox\" name=\"rpt_sun\" value=\"y\\\" . (!empty($rpt_sun)? \" checked=\"checked\\\":\"\") . \" \"/>&nbsp; " . translate("Sunday") . "</label>\n"; | | |

```

```

?></td></tr>
</table>

<?php if ( $useTabs ) { ?>
</div> <!-- End tabscontent_pete -->
<?php } /* $useTabs */ ?>
<?php } ?>
</div> <!-- End tabscontent -->
<table style="border-width:0px;">
<tr><td>
    <script type="text/javascript">
<!-- <![CDATA[
        document.writeln      (      '<input      type="button"      value="<?php
etranlate("Save")?>" onclick="validate_and_submit()" />' );
//]]> -->
    </script>
    <noscript>
        <input type="submit" value="<?php etranlate("Save")?>" />
    </noscript>
<br>
<br>
<br>
</td></tr>
</table>
<input type="hidden" name="participant_list" value="" />
</form>

<?php if ( $id > 0 && ( $login == $create_by || $single_user == "Y" ||
$is_admin ) ) { ?>
    <a href="del_entry.php?id=<?php echo $id;?>" onclick="return
confirm('<?php etranlate("Are you sure you want to delete this
entry?")?>');"><?php etranlate("Delete entry")?></a><br />
<?php
    } //end if clause for delete link
} else {
    echo translate("You are not authorized to edit this entry") . ".";
} //end if ( $can_edit )
?>

<?php print_trailer(); ?>
</body>
</html>

```

D. TEST.PHP

```

<?php
include_once 'includes/init.php';
require_once('../Connections/IMASInternalDatabase.php');

//gets email info from database
$query_Get_Email = "SELECT email_name, email_password, email_imap
FROM email_account_tracking WHERE cal_login='$login' LIMIT 1;";
$Get_Email = mysql_query($query_Get_Email, $IMASInternalDatabase)
or die(mysql_error());
$row_Get_Email = mysql_fetch_assoc($Get_Email);

```



```

        //gets cellphone number and sms gateway from database
        $query_Get_cellphone = "SELECT cellphone_number, sms_gateway FROM
cellphone_account_tracking, cell_provider WHERE cal_login='$login'
        &&
cellphone_account_tracking.cell_provider=cell_provider.cell_provider
LIMIT 1;";
        $Get_cellphone = mysql_query($query_Get_cellphone,
$IMASInternalDatabase) or die(mysql_error());
        $row_Get_cellphone = mysql_fetch_assoc($Get_cellphone);

        //create email and sms gateway variables
        $user_login=$row_Get_Email['email_name'];
        $user_password= $row_Get_Email['email_password'];
        $user_imap=$row_Get_Email['email_imap'];
        $user_cellnumber=$row_Get_cellphone['cellphone_number'];
        $user_sms_gateway=$row_Get_cellphone['sms_gateway'];

        //Open the users email mailbox
        $mailbox = imap_open("{'.$user_imap.':143}INBOX", $user_login,
$user_password); //This grabs e-mail account(IMAP only) info from
database

        // Check messages
        $check = imap_check($mailbox);

        $NumberOfMessages = $check->Nmsgs;
        $NumberOfRecentMessages = $check->Recent;

        print("<PRE>");
        print("Connected to " . $check->Mailbox);
        print("<BR>");
        print("Date most recent message : " . $check->Date);
        print("<BR>");
        //print("Connection type : " . $check->Driver);
        //print("<BR>");
        //print("Name of the mailbox : " . $check->Mailbox);
        //print("<BR>");
        print("Total number of messages : " . $NumberOfMessages);
        print("<BR>");
        print("Number of recent messages : " . $NumberOfRecentMessages);
        print("<BR>");
        print("</PRE>");

        if ($NumberOfRecentMessages > 0) {
            $InitialMessageNumber = ($NumberOfMessages) -
($NumberOfRecentMessages) + 1;
            echo "Initial Message Number = " . $InitialMessageNumber .
"<br/>";

            for($InitialMessageNumber; $InitialMessageNumber <=
$NumberOfMessages; $InitialMessageNumber++)
            {
                echo "Current Message Number = " .
$InitialMessageNumber . "<br/>";
                $mailHeader = imap_headerinfo($mailbox,
$InitialMessageNumber);
            }
        }
    }
}

```

```

        $from = $mailHeader->fromaddress;
        $subject = strip_tags($mailHeader->subject);
        $date = $mailHeader->date;
        echo "From: $from";
        echo "<br/>";
        echo "Subject: $subject";
        echo "<br/>";
        echo "Date: $date";
        echo "<br/>";
        $messageBody = imap_fetchbody($mailbox,
$InitialMessageNumber, '1');
        echo "Message: <br/>";
        echo $messageBody;
        echo "<hr>";

        //Send the new messages to SMS gateway
        $sendto = $user_cellnumber . "@" . $user_sms_gateway;
        echo $sendto;
        echo "<br/>";
        mail($sendto, $subject, $messageBody);
        } //This sends the SMS message
    }
    else { echo "No new messages!<br/>";}
    imap_close($mailbox);
?>

```

E. DATABASE TABLES

Table structure for table action- check all the way through

Field	Type	Null	Default
<i>cal_access</i>	char(1)	Yes	
<i>cal_priority</i>	int(1)	Yes	
action_description	set('cellphone message', 'IM', 'email', 'cellphone call')	Yes	

Table structure for table alert_store

Field	Type	Null	Default
User_login	varchar(25)	Yes	
<i>Media_store_ID</i>	mediumint(32)	Yes	NULL
Date_time	datetime	Yes	
Type	enum('email', 'text', 'IM')	Yes	
Filepath	varchar(32)	Yes	

Table structure for table cell_provider

Field	Type	Null	Default
<i>cell_provider</i>	varchar(25)	Yes	
sms_gateway	varchar(25)	Yes	

Table structure for table cellphone_account_tracking

Field	Type	Null	Default
<i>cellphone_account_ID</i>	bigint(32)	Yes	NULL
cellphone_number	varchar(10)	Yes	
cal_login	varchar(25)	Yes	
device_brand_number	varchar(25)	Yes	
cell_provider	varchar(25)	Yes	

Table structure for table device

Field	Type	Null	Default
<i>device_brand_number</i>	varchar(25)	Yes	
device_specs	varchar(25)	Yes	

Table structure for table email_account_tracking

Field	Type	Null	Default
<i>Email_address</i>	varchar(25)	Yes	
cal_login	varchar(25)	Yes	
Email_password	varchar(32)	Yes	
Email_imap	varchar(32)	Yes	
Email_name	varchar(25)	Yes	

Table structure for table im_account_tracking

Field	Type	Null	Default
<i>Screen_name</i>	varchar(25)	Yes	
<i>IM_provider</i>	varchar(25)	Yes	
cal_login	varchar(25)	Yes	

Table structure for table im_provider

Field	Type	Null	Default
<i>IM_provider</i>	varchar(25)	Yes	

Table structure for table news

Field	Type	Null	Default
<i>News_ID</i>	bigint(32)	Yes	NULL
Title	varchar(255)	Yes	
Description	text	Yes	
Date	datetime	Yes	
Private	set('no', 'yes')	Yes	no

Table structure for table webcal_asst

Field	Type	Null	Default
<i>cal_boss</i>	varchar(25)	Yes	
<i>cal_assistant</i>	varchar(25)	Yes	

Table structure for table webcal_categories

Field	Type	Null	Default
<i>cat_id</i>	int(11)	Yes	
cat_owner	varchar(25)	Yes	NULL
cat_name	varchar(80)	Yes	

Table structure for table webcal_config

Field	Type	Null	Default
cal_setting	varchar(50)	Yes	
cal_value	varchar(100)	Yes	NULL

Table structure for table webcal_entry

Field	Type	Null	Default
cal_id	int(11)	Yes	
cal_group_id	int(11)	Yes	NULL
cal_ext_for_id	int(11)	Yes	NULL
cal_create_by	varchar(25)	Yes	
cal_date	int(11)	Yes	
cal_time	int(11)	Yes	NULL
cal_mod_date	int(11)	Yes	NULL
cal_mod_time	int(11)	Yes	NULL
cal_duration	int(11)	Yes	
cal_priority	int(11)	Yes	2
cal_type	char(1)	Yes	E
cal_access	char(1)	Yes	P
cal_name	varchar(80)	Yes	
cal_description	text	Yes	NULL

Table structure for table webcal_entry_ext_user

Field	Type	Null	Default
cal_id	int(11)	Yes	0
cal_fullname	varchar(50)	Yes	
cal_email	varchar(75)	Yes	NULL

Table structure for table webcal_entry_log

Field	Type	Null	Default
cal_log_id	int(11)	Yes	
cal_entry_id	int(11)	Yes	
cal_login	varchar(25)	Yes	
cal_user_cal	varchar(25)	Yes	NULL
cal_type	char(1)	Yes	
cal_date	int(11)	Yes	
cal_time	int(11)	Yes	NULL
cal_text	text	Yes	NULL

Table structure for table webcal_entry_repeats

Field	Type	Null	Default
<i>cal_id</i>	int(11)	Yes	0
cal_type	varchar(20)	Yes	NULL
cal_end	int(11)	Yes	NULL
cal_frequency	int(11)	Yes	1
cal_days	char(7)	Yes	NULL

Table structure for table webcal_entry_repeats_not

Field	Type	Null	Default
<i>cal_id</i>	int(11)	Yes	
<i>cal_date</i>	int(11)	Yes	

Table structure for table webcal_entry_user

Field	Type	Null	Default
<i>cal_id</i>	int(11)	Yes	0
<i>cal_login</i>	varchar(25)	Yes	
cal_status	char(1)	Yes	A
cal_category	int(11)	Yes	NULL

Table structure for table webcal_group

Field	Type	Null	Default
<i>cal_group_id</i>	int(11)	Yes	
cal_owner	varchar(25)	Yes	NULL
cal_name	varchar(50)	Yes	
cal_last_update	int(11)	Yes	

Table structure for table webcal_group_user

Field	Type	Null	Default
<i>cal_group_id</i>	int(11)	Yes	
<i>cal_login</i>	varchar(25)	Yes	

Table structure for table webcal_import

Field	Type	Null	Default
<i>cal_import_id</i>	int(11)	Yes	
cal_name	varchar(50)	Yes	NULL
cal_date	int(11)	Yes	
cal_type	varchar(10)	Yes	
cal_login	varchar(25)	Yes	NULL

Table structure for table webcal_import_data

Field	Type	Null	Default
cal_import_id	int(11)	Yes	
cal_id	int(11)	Yes	
cal_login	varchar(25)	Yes	
cal_import_type	varchar(15)	Yes	
cal_external_id	varchar(200)	Yes	NULL

Table structure for table webcal_nonuser_cals

Field	Type	Null	Default
cal_login	varchar(25)	Yes	
cal_lastname	varchar(25)	Yes	NULL
cal_firstname	varchar(25)	Yes	NULL
cal_admin	varchar(25)	Yes	

Table structure for table webcal_reminder_log

Field	Type	Null	Default
cal_id	int(11)	Yes	0
cal_name	varchar(25)	Yes	
cal_event_date	int(11)	Yes	0
cal_last_sent	int(11)	Yes	0

Table structure for table webcal_report

Field	Type	Null	Default
cal_login	varchar(25)	Yes	
cal_report_id	int(11)	Yes	
cal_is_global	char(1)	Yes	N
cal_report_type	varchar(20)	Yes	
cal_include_header	char(1)	Yes	Y
cal_report_name	varchar(50)	Yes	
cal_time_range	int(11)	Yes	
cal_user	varchar(25)	Yes	NULL
cal_allow_nav	char(1)	Yes	Y
cal_cat_id	int(11)	Yes	NULL
cal_include_empty	char(1)	Yes	N
cal_show_in_trailer	char(1)	Yes	N
cal_update_date	int(11)	Yes	

Table structure for table webcal_report_template

Field	Type	Null	Default
cal_report_id	int(11)	Yes	
cal_template_type	char(1)	Yes	
cal_template_text	text	Yes	NULL

Table structure for table webcal_site_extras

Field	Type	Null	Default
cal_id	int(11)	Yes	0
cal_name	varchar(25)	Yes	
cal_type	int(11)	Yes	
cal_date	int(11)	Yes	0
cal_remind	int(11)	Yes	0
cal_data	text	Yes	NULL

Table structure for table webcal_user

Field	Type	Null	Default
cal_login	varchar(25)	Yes	
cal_passwd	varchar(32)	Yes	NULL
cal_lastname	varchar(25)	Yes	NULL
cal_firstname	varchar(25)	Yes	NULL
cal_is_admin	char(1)	Yes	N
cal_email	varchar(75)	Yes	

Table structure for table webcal_user_layers

Field	Type	Null	Default
cal_layerid	int(11)	Yes	0
cal_login	varchar(25)	Yes	
cal_layeruser	varchar(25)	Yes	
cal_color	varchar(25)	Yes	NULL
cal_dups	char(1)	Yes	N

Table structure for table webcal_user_pref

Field	Type	Null	Default
cal_login	varchar(25)	Yes	
cal_setting	varchar(25)	Yes	
cal_value	varchar(100)	Yes	NULL

Table structure for table webcal_view

Field	Type	Null	Default
<i>cal_view_id</i>	int(11)	Yes	
<i>cal_owner</i>	varchar(25)	Yes	
<i>cal_name</i>	varchar(50)	Yes	
<i>cal_view_type</i>	char(1)	Yes	NULL
<i>cal_is_global</i>	char(1)	Yes	N

Table structure for table webcal_view_user

Field	Type	Null	Default
<i>cal_view_id</i>	int(11)	Yes	
<i>cal_login</i>	varchar(25)	Yes	

LIST OF REFERENCES

- Apache. 2006. *How the ASF works - The Apache Software Foundation*. Internet on-line. Available from <http://www.apache.org/foundation/how-it-works.html#history>>. [Accessed April 13, 2006].
- CPAN. 2006. *CPAN/src*. Internet on-line. Available from <http://www.cpan.org/src/README.html>>. [Accessed April 16, 2006].
- Dayem, Rifaat A. 1997. *Mobile Data & Wireless LAN Technologies*. Upper Saddle River, NJ: Prentice Hall.
- Deitel, Harvey M., Paul J. Deitel, and Andrew B. Goldberg. 2004. *Internet & World Wide Web: How to Program. Third Edition*. Upper Saddle River, NJ: Pearson/Prentice Hall.
- GSM World. 2006. *GSM World - Frequently Asked Questions*. Internet on-line. Available from <http://www.gsmworld.com/technology/faq.shtml>>. [Accessed April 20, 2006].
- GSM World. 2006. *GSM World - GSM Technology*. Internet on-line. Available from <http://www.gsmworld.com/technology/index.shtml>>. [Accessed April 20, 2006].
- Insurance Information Institute. 2006. *Cell Phones and Driving*. Internet on-line. Available from <http://www.iii.org/media/hottopics/insurance/cellphones/>>. [Accessed April 25, 2006].
- Kroenke, David M. 2004. *Database Processing*. 9th Ed. Upper Saddle River, NJ: Prentice Hall.
- Microsoft. 2006. *Internet Information Services 6.0 Features*. Internet on-line. Available from <http://www.microsoft.com/windowsserver2003/iis/evaluation/features/default.aspx>>. [Accessed April 13, 2006].

Microsoft. 2006. *Microsoft SQL Server 2005 Overview: Microsoft TechNet SQL Server TechCenter*. Internet on-line. Available from <http://www.microsoft.com/sql/prodinfo/overview/default.aspx>. [Accessed April 15, 2006].

Microsoft. 2006. *Windows Products and Technologies History: Windows Server Products History*. Internet on-line. Available from <http://www.microsoft.com/windows/WinHistoryServer.aspx>. [Accessed April 13, 2006].

Muller, Nathan J. 2000. *Desktop Encyclopedia of Telecommunications Second Edition*. New York, NY: McGraw-Hill.

MySQL. 2006. *MySQL 5.0 Reference Manual :: 1.4 Overview of the MySQL Database Management System*. Internet on-line. Available from <http://dev.mysql.com/doc/refman/5.0/en/what-is.html>. [Accessed April 13, 2006].

Netcraft. 2006. *Netcraft: Web Server Survey Archive*. Internet on-line. Available from http://news.netcraft.com/archives/web_server_survey.html. [Accessed April 13, 2006].

Oracle. 2006. *Oracle Database and Grids*. Internet on-line. Available from <http://www.oracle.com/database/index.html>. [Accessed April 15, 2006].

Oracle. 2006. *Oracle Grid Computing*. Internet on-line. Available from <http://www.oracle.com/technologies/grid/index.html>. [Accessed April 15, 2006].

Perl. 2006. *About Perl - perl.org*. Internet on-line. Available from <http://www.perl.org/about.html>. [Accessed April 16, 2006].

PHP. 2006. *PHP: General Information - Manual*. Internet on-line. Available from <http://us2.php.net/manual/en/faq.general.php>. [Accessed April 17, 2006].

PHP EasyWindows Installer. 2005. *PHP EasyWindows Installer PHP Everywhere*. Internet on-line. Available from <<http://phplens.com/phpeverywhere/easywindows>>. [Accessed April 17, 2006].

PostgreSQL. 2006. *PostgreSQL: Frequently Asked Questions(FAQ) for PostgreSQL*. Internet on-line. Available from <<http://www.postgresql.org/docs/faqs.FAQ.html>>. [Accessed April 13, 2006].

Schwartz, Mischa. 2005. *Mobile Wireless Communications*. Cambridge, UK: Cambridge University Press.

Ward, Tyrone. 2001. *A Database of Adversary Decision Makers*. Master's Thesis, Naval Postgraduate School.

Whitten, Jeffrey L., Bentley, Lonnie D., Dittman, Kevin C. 2004. *Systems Analysis and Design Methods*. 6th Ed. New York, NY: McGraw Hill.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, Virginia 22060-6218
2. Dudley Knox Library
Naval Postgraduate School
411 Dyer Road 93943-5101
Monterey, California
3. Dean Dan Boger, Code IW
Naval Postgraduate School
Root Hall
Monterey, CA 93943-5118